NASA CR-158,840

NASA-CR-158840
19790019965

A Reproduced Copy

OF

Reproduced for NASA

by the

NASA Scientific and Technical Information Facility

FFNo 672 Aug 65

COO-3077-158

Courant Mathematics and Computing Laboratory

U. S. Department of Energy

(NASA-CR-158840) AN ARTIFICIAL VISCOSITY METHOD FOR THE DESIGN OF SUPERCRITICAL AIRFOILS (New York Univ.) 168 p HC A08/MF A01 CSCL 01A

N79-28136

Unclas
G3/02 29406

An Artificial Viscosity Method for the Design of Supercritical Airfoils

Geoffrey B. McFadden

Research and Development Report

Prepared under Contract EY-76-C-02-3077 with the Office of Energy Research; NASA-Ames Research Center Grant NGR-33-016-201; NASA-Langley Research Center Grant NSG 1579; and the National Science Foundation

Mathematics and Computing
July 1979

New York University

UNCLASSIFIED

Courant Mathematics and Computing Laboratory

New York University

Mathematics and Computing COO-3077-158

An Artificial Viscosity Method for the Design of Supercritical Airfoils

Geoffrey B. McFadden

July 1979

Prepared under Contract EY-76-C-02-3077
with the Office of Energy Research;
NASA-Ames Research Center Grant NGR-33-016-201;
NASA-Langley Research Grant NSG 1579; and the
National Science Foundation

UNCLASSIFIED

Table of Contents

Preface

The need for increased efficiency in the use of our energy resources has stimulated applied research in many areas. Recently progress has been made in the field of aerodynamics, where the development of the supercritical wing promises significant savings in the fuel consumption of aircraft operating near the speed of sound. Computational transonic aerodynamics has proved to be a useful tool in the design and evaluation of these wings.

We present here a numerical technique for the design of two-dimensional supercritical wing sections with low wave drag. The method is actually a design mode of the analysis code H developed by Bauer, Garabedian, and Korn [2,3,4]. This analysis code gives excellent agreement with experimental results and is used widely by the aircraft industry. We hope the addition of a conceptually simple design version will make this code even more useful to the engineering public.

I. INTRODUCTION

1. Description of the Problem

In this section we discuss some of the principles behind the supercritical wing and describe our contribution to the subject.

General considerations show that the range of an aircraft is roughly proportional to the parameter $M_\infty L/D$, where L is the lift, D is the drag, and the free stream Mach number M_∞ is the ratio of the aircraft's speed to the speed of sound. The top curve in Figure 1 shows the general behavior of this parameter as the Mach number M_∞ is varied. The value of $M_\infty L/D$ that maximizes the range of the aircraft occurs near a region of rapid increase in drag known as drag rise, shown by the bottom curve of Figure 1. At this speed the flow is observed to be transonic, with regions of supersonic flow appearing where the air accelerates over the wing. When the free stream Mach number has the value M_C depicted in Figure 1, the maximum speed of the flow is equal to the speed of sound. When $M_\infty > M_C$ the flow is said to be supercritical.

Figure 2 illustrates some important characteristics of the flow past an airfoil at speeds corresponding to drag rise. The region of supersonic flow is terminated by a shock, where the pressure is observed to be discontinuous.

As the speed of the wing is increased, the supersonic zone grows in size and the pressure jump becomes larger. The occurrence of such shocks in the flow imposes a retarding force on the wing known as wave drag, which is one reason for the drag rise seen in Figure 1. The large pressure gradients present in strong shocks can also induce separation of the boundary layer of air that adheres to the wing because of friction; separation results in a decrease in lift and more drag. The study of transonic flow is therefore important not only because transonic flow encompasses the most economical regime for aircraft operation, but because the deterioration of an aircraft's efficiency at higher speeds is due to transonic effects.

The supercritical wing is designed to delay the onset of drag rise to higher Mach numbers. Since the efficiency of the wing is governed by the optimal value of $M_\infty L/D$, postponing the onset of drag rise to higher Mach numbers results in a corresponding decrease in the fuel requirements of the aircraft. The delay in drag rise can be effected by constructing the wing so that the strong shocks accompanying supersonic zones of moderate size on conventional wings are replaced by weaker shocks with less wave drag and no appreciable boundary layer separation.

We are mainly concerned here with the contributions to supercritical wing technology made by computational transonic aerodynamics. The numerical solution of the partial

differential equations of gas dynamics provides a theoretical means for both the design and evaluation of supercritical wings. For example, two-dimensional shockless airfoils can be obtained by calculating real analytic solutions to the hodograph equations of transonic flow. These airfoils have the property that at a specified speed and angle of attack, the calculated two-dimensional transonic flow is smooth. This guarantees that the wave drag will be small near at least one operating condition. It may happen that there is an increase in wave drag at supercritical Mach numbers below the design condition known as drag creep. Since a wing must operate efficiently over a range of conditions it is desirable to avoid the occurrence of drag creep. Provided this is done, a practical approach to the supercritical wing is to design the wing using computer-generated shockless airfoils in each cross-section.

It is also possible to evaluate the performance of wings at off-design conditions using computer codes. Programs that calculate the three-dimensional transonic flow past wing-body combinations are used regularly by the aircraft industry. Considerable savings can be realized by replacing preliminary wind tunnel testing of new wing designs by such computer simulation.

There is presently much interest in the possibility of developing a numerical scheme for the design of three-dimensional wing-body combinations. Hodograph methods employed

in the design of two-dimensional shockless wing sections cannot be used for this purpose. Our contribution in this direction is a two-dimensional design code based on techniques that may prove useful in such an endeavor. Although our method does not produce shockless airfoils, we show that it is possible to obtain wing sections with low wave drag by using an artificial viscosity to smear shocks properly. The design procedure is outlined in Section 4.1.

Our program is actually a new design mode of the two-dimensional analysis code H developed by Bauer, Garabedian, and Korn [2,3,4]. This analysis code solves the direct problem of obtaining the transonic flow past a given wing section. The code includes a turbulent boundary layer correction which gives a reliable approximation to the drag due to skin friction and predicts boundary layer separation. Drag estimates obtained with the code are in good agreement with experiment, and the program has found wide acceptance in the aircraft industry.

There are two major steps in the operation of the analysis routine. Given the coordinates of the airfoil, the region exterior to the airfoil is mapped conformally to the interior of the unit circle as in Sell's treatment of subcritical flow past an airfoil [33]. The nonlinear partial differential equations of transonic flow are then solved iteratively in the unit circle using a type-dependent difference scheme similar to the one first

developed by Murman and Cole [27]. If boundary layer corrections are desired, the shock wave - boundary layer interaction is simulated by iterating between inviscid flow calculations and boundary layer corrections until convergence is achieved.

The design modification we have added to the code solves the inverse problem of calculating the shape an airfoil must have in order to achieve a specified pressure distribution. When operating in this mode, an initial guess is provided for the shape of the desired airfoil and the region exterior to this airfoil is mapped into the unit circle as in the analysis mode. A number of flow iterations are performed using an artificial viscosity that inhibits the formation of shocks, as described in Section 2.2. The pressure distribution resulting from these calculations is then compared with the desired input pressure distribution and a better approximation to the desired airfoil is obtained, as described in Section 2.3. This new profile is mapped to the unit circle as in the first step and the process is repeated until the approximations converge. A boundary layer correction may then be calculated on the basis of the last pressure distribution. The desired airfoil is obtained by subtracting the displacement thickness of the turbulent boundary layer from the coordinates of the computed profile.

The inverse method transfers the difficulty in designing wings from determining the coordinates of the airfoil to finding pressure distributions that give rise to airfoils with desired specifications. We therefore include descriptions of some pressure distributions that generate airfoils with low wave drag, and indicate how to modify the input distribution in order to obtain airfoils with a desired lift, thickness-to-chord ratio, and design Mach number.

The remainder of the paper is organized as follows. The mathematical statement of the problem is formulated in Chapter II. The computational procedure is outlined in Chapter III. In Chapter IV we present results obtained with the design mode, together with some comparisons to airfoils obtained by other methods. Chapter V is more theoretical in nature and includes a convergence proof for the design problem in a special case of subsonic flow. We provide a description of the modified version of the Bauer, Garabedian, and Korn analysis code H in Chapter VI.

I would like to express my gratitude for the advice and encouragement of Paul R. Garabedian, who suggested this problem and made the work possible. I am also grateful for the help of Frances Bauer and Antony Jameson at various stages of the research, and for a fast and accurate typing job by Connie Engle.

2. References to Other Work

Transonic flow research has a colorful history [5,29]. In the late 1940's, arguments to the effect that smooth transonic flows past arbitrary profiles should not generally be expected to exist were formulated by Busemann [10], Frankl [15], and Guderley [16]. These observations raised doubts about the physical significance of the smooth solutions to the steady, two-dimensional potential equation for transonic flow that were known at the time. It was observed experimentally that transonic flows generally exhibit shocks when the supersonic zones are of moderate size, but there were occasional instances of near-shockless flow that seemed to contradict the implications of the nonexistence theorems. A "transonic controversy" developed over the true nature of transonic flows in general and of shockless flows in particular.

The controversy attracted considerable attention from mathematicians in the hopes that a rigorous investigation of whether the flow problem was well-posed would help clarify matters. From this viewpoint, a satisfactory demonstration that the problem of finding smooth transonic flows past convex symmetric profiles was not correctly set was supplied by Morawetz [26], although the apparent discrepancies between theory and experiment remained unresolved.

More progress was made in the early 1960's with the experimental work of Pearcey [31], who was able to systemati-

cally produce near-shockless flows past wing sections having a suction pressure peak near the nose of the profile. The subsequent development of numerical techniques capable of treating transonic flows with shocks brought about a reinterpretation of the nonexistence theorem since the computational problem seems to be correctly set in terms of weak solutions. Results of both experiment and computation show that shockless and neighboring near-shockless solutions do in fact have physical significance and can provide an important means of reducing the drag experienced by airfoils travelling at transonic speeds.

Several numerical techniques have been developed for the design of two-dimensional supercritical wing sections using inviscid flow theory. We can distinguish between approaches relying on hodograph methods and the remaining approaches.

The hodograph transformation consists of reversing the roles of the dependent and independent variables in the flow equations with the result that the partial differential equations are linear. Using this transformation, several methods have been devised to allow the systematic computation of airfoils that have shockless flows at given operating conditions [2,4,8,30]. Such airfoils necessarily have low wave drag at nearby operating conditions, although drag creep can occur elsewhere.

Other approaches to the design problem do not generally provide shock-free solutions to the equations. This is not necessarily a disadvantage, since for some applications it is possible that an airfoil designed with a weak shock might have an overall performance that is as good or better than a shockless airfoil with similar specifications. The main difficulty is to find pressure distributions that will generate airfoils with low drag levels.

Some design methods use the approximations of small disturbance theory and thin airfoil theory [11,22,31]. With this approach the solution is expanded in terms of a parameter describing the thickness of the profile, which is assumed to be small. This has the advantage that to leading order the profile can be replaced by a given slit. The desired pressure distrubution along the surface of the airfoil can then be used in a boundary condition applied at the slit, and so the difficulties caused by the unknown boundary are avoided. The coordinates of the desired airfoil can be determined from the resulting velocity components. This technique has the disadvantage that the flow is not represented correctly near the stagnation point at the leading edge of a blunt-nosed airfoil. It should be noted that applications of this technique to the design of three-dimensional wings have been initiated [17,34].

The design methods of Carlson [12] and Tranen [36] solve the inverse problem for the full potential equation with a free boundary. Both of these methods use the prescribed pressure distribution in a boundary condition for the determination of the velocity potential, and calculate the position of the surface of the profile by using the condition of flow tangency along the body. In Carlson's method the calculation is performed using Cartesian coordinates. The coordinates near the nose are given in advance and the remainder of the profile is determined as a free boundary. Tranen uses the analysis code H to perform the flow calculations and to provide a computational domain, and proceeds by alternating between analysis and design computations. At each cycle the user modifies the prescribed pressure distribution in order to achieve convergence.

The design procedure of Hicks and his associates [18] is based on the use of a numerical optimization routine together with the analysis code H to minimize the drag coefficient with respect to design variables that describe the shape of the profile, while satisfying various constraints on the operating conditions and geometry. This technique has the advantage of drag reduction without the necessity of choosing the pressure distribution. Its main drawback is the large amount of computing time required when many parameters are allowed to vary.

The method we present for supercritical wing design uses the prescribed pressure distribution in a boundary condition for the determination of the conformal mapping from the unit circle to the desired airfoil. This approach to the design problem is similar in spirit to Lighthill's inverse method [23], which is based on the linear theory of incompressible flow and so does not require an iterative procedure to determine the flow and profile. Other incompressible treatments along these lines have also appeared [1,13].

The practical success of an inverse method of airfoil design depends on the prescribed pressure distribution. It is therefore important to study the relation between the assigned pressure distribution and the performance of the resulting airfoil [7,29]. Much work remains to be done on this aspect of the problem. The many shockless flows produced by hodograph methods provide a good basis for investigation.

II. THE PARTIAL DIFFERENTIAL EQUATIONS OF TRANSONIC FLOW

In this chapter we consider the mathematical formulation of the design problem. We summarize the basic equations of motion for gas dynamics and discuss the boundary conditions appropriate for the direct and inverse problems.

1. The Equations of Gas Dynamics

We begin with some comments about the choice of equations to describe the problem. We wish to model the flight of aerodynamically efficient wings at transonic speeds. We are especially concerned with the drag on such bodies, which includes forces due to skin friction and shocks. For our treatment to be of practical use we must consider equations which allow estimates of the wave drag due to shocks, and we must provide for the calculation of viscous effects. Furthermore, we must choose equations that are compatible with the inherent storage limitations of computers.

It is common in experiment as well as theory to treat the case of steady, two-dimensional flow past a wing of uniform cross section. This provides a good approximation of the flow near the middle section of a three-dimensional wing with a straight leading edge

moving with a constant velocity. This geometrical simplification permits the use of just two independent variables, which we take to be the x and y coordinates.

Another important simplification is possible if the airfoil is streamlined so that viscous effects are confined to the immediate vicinity of the profile. In this case the flow outside the boundary layer can be obtained from lower order partial differential equations describing inviscid fluid motion. The inviscid solution can then be used to calculate a boundary layer correction to the flow past the airfoil, making it possible to obtain estimates of the drag due to skin friction [28,32]. Separation of the boundary layer should be avoided for aerodynamical reasons, too, so it is important for the theory to give reliable estimates of the growth of the boundary layer.

Inviscid fluid flow can be described by conservation laws consisting of nonlinear, first order partial differential equations involving the velocity components of the flow and two thermodynamic variables such as the density and entropy. The conservation laws also provide shock conditions which determine the jump in these quantities across a surface of discontinuity in the flow. For the case of flows past thin bodies at speeds close to the speed of sound, the shocks are usually weak in the sense that the jump in velocity across the shock is small compared to the speed of sound. The jump in entropy across a shock

is of third order in the shock strength; to a good approximation, the change in entropy can therefore be neglected in a weak shock. With this assumption, the entropy of a fluid particle is constant throughout its motion, and the flow may be considered isentropic.

As a result of considering the entropy to be conserved across a shock, the shock condition expressing conservation of the normal component of momentum is lost. The defect in this quantity can be interpreted as an approximation of the wave drag exerted on the airfoil by the shock.

For isentropic flow, the velocity field is irrotational if the flow is uniform at infinity. This permits the introduction of a velocity potential whose derivatives are the velocity components. The inviscid equations of motion can then be reduced to a single second order partial differential equation for the potential, and in the computation it is only necessary to store values of a single dependent variable.

We shall list below the equations describing the flow of an inviscid, isentropic gas. In Section 3.4 the equations used to describe a turbulent boundary layer correction will be discussed. It is found in practice that these equations provide a description of the transonic flow past an airfoil that agrees well with experiment over a wide range of conditions [3,4].

The equations describing the steady two-dimensional motion of an ideal polytropic gas are familiar [14,25]. From thermodynamics we have the equation of state

(2.1) $$p = A(s)\rho^{\gamma}$$

where p, ρ, and S are the pressure, density, and specific entropy of the gas. $A(s)$ is a known function of the entropy and $\gamma > 1$ is a constant depending on the nature of the gas.

Conservation of mass gives

(2.2) $$(\rho u)_x + (\rho v)_y = 0$$

where u and v are the x and y components of the velocity.

Similarly, the conservation of momentum asserts that

(2.3) $$(uu_x + vu_y) + p_x = 0 ,$$

(2.4) $$(uv_x + vv_y) + p_y = 0 ,$$

and the conservation of energy gives

(2.5) $$uS_x + vS_y = 0 .$$

As mentioned above, we consider the case of constant entropy, so that (2.5) is automatically satisfied and $A(S)$ in (2.1) is a constant. The flows we consider become uniform at large distances. It then follows from a theorem of Kelvin that the flow is irrotational,

(2.6) $$u_y - v_x = 0 \ .$$

Using (2.1), (2.2) and (2.6) we obtain Bernoulli's law

(2.7) $$\frac{u^2 + v^2}{2} + \frac{c^2}{\gamma-1} = \frac{1}{2} \ \frac{\gamma + 1}{\gamma - 1} \ c_*^2 \ ,$$

where $c^2 = dp/d\rho$ is the square of the local speed of sound and c_* is a constant known as the critical speed. The dimensionless ratio

$$M = \left(\frac{u^2 + v^2}{c^2}\right)^{1/2}$$

is called the local Mach number and is greater than one if $u^2 + v^2 = q^2 > c_*^2$ (locally supersonic flow) and less than one if $q^2 < c_*^2$ (locally subsonic flow).

According to (2.2) and (2.6), there are two functions ϕ and ψ such that

(2.8a) $$u = \phi_x = \psi_y/\rho$$

(2.8b) $$v = \phi_y = -\psi_x/\rho$$

ϕ and ψ are the velocity potential and stream function of the flow. We may obtain a single equation for ϕ from (2.1), (2.2), and (2.7),

(2.9) $$(c^2 - \phi_x{}^2)\phi_{xx} - 2\phi_x\phi_y\phi_{xy} + (c^2 - \phi_y{}^2)\phi_{yy} = 0 \ .$$

This is the partial differential equation that is the basis of our numerical work. ψ satisfies a similar equation.

Equation (2.9) is a quasilinear partial differential equation which is elliptic when $M^2 < 1$ and hyperbolic when $M^2 > 1$. We are interested in the case of transonic flow, so that (2.9) has mixed type in the region of interest. It is appropriate to consider weak solutions to (2.9) or (2.2) under the conditions that ϕ is continuous and that any shocks present are compressive. This corresponds to the proper entropy inequality in nonisentropic flows.

2. The Direct Problem

In this section we discuss the formulation of the direct, or analysis, problem of determining the flow past a given wing section.

We consider an airfoil with coordinates $(x(s),y(s))$ parametrized by arc length s measured from tail to tail as in Figure 2. The included angle at the trailing edge is denoted by ε. The frame of reference is chosen so that the airfoil is at rest and the air has a resulting velocity $\underset{\sim}{u} = (q_\infty \cos\alpha,\ q_\infty \sin\alpha)$ at infinity, where the angle of attack α gives the direction of motion relative to fixed coordinate axes.

The fact that the flow must be tangential to the surface of the airfoil provides one boundary condition for (2.9). If $\hat{\nu}$ is a unit normal to the profile, then this condition can be stated as

$$\hat{\nu}\cdot\underset{\sim}{u} = \frac{\partial\phi}{\partial\nu} = 0 \tag{2.10}$$

on the curve $(x(s),y(s))$. Since the airfoil is a streamline of the flow, this fact can also be expressed by

$$\psi\big(x(s),y(s)\big) = \text{constant}. \tag{2.11}$$

We consider lifting profiles with cusped trailing edges, $0 \le \varepsilon << 1$, in which case the potential ϕ need not be single-valued. The circulation $\Gamma = [\phi]$ of the flow around the airfoil is then uniquely determined by the Kutta-Joukowski condition

$$(2.12) \qquad |\underset{\sim}{u}(x(0),y(0))| < \infty ,$$

which states that the velocity must be finite at the trailing edge. If $\varepsilon > 0$ the flow necessarily has a stagnation point at the trailing edge. This is not the case for $\varepsilon = 0$. The presence of a stagnation point at the tail of the airfoil should generally be avoided since the resulting adverse pressure gradient promotes separation of the boundary layer.

It can be shown [24] that ϕ has an asymptotic expansion

$$(2.13) \qquad \phi \sim q_\infty r \cos(\theta-\alpha) + \frac{\Gamma}{2\pi} \tan^{-1}(\beta \tan(\theta-\alpha))$$

as $r^2 = x^2 + y^2 \to \infty$, where $\beta^2 = 1 - M_\infty^2$ and $\theta = \tan^{-1} y/x$. This is similar to the corresponding expansion for incompressible flow. The Prandtl-Glauert scale factor β commonly occurs in linearized treatments of compressible flow.

For the analysis problem it is convenient to express Bernoulli's law (2.7) in the form

$$(2.14) \qquad \frac{\phi_x^2+\phi_y^2}{2} + \frac{c^2}{\gamma-1} = q_\infty^2\left(\frac{1}{2} + \frac{1}{(\gamma-1)M_\infty^2}\right),$$

where M_∞ is the Mach number at infinity. With this notation, the direct problem can be formulated by specifying the airfoil coordinates $(x(s),y(s))$, the angle of attack α,

and $M_\infty < 1$. In this case we are free to choose the units so that q_∞ is normalized to one. The flow is then obtained by solving the partial differential equation (2.9), along with the boundary conditions (2.10), (2.12), (2.13), and (2.14).

The lift of the airfoil is proportional to the circulation Γ. When the angle of attack is varied, the circulation of the flow adjusts so that the Kutta condition (2.12) is satisfied. For a given Mach number M_∞, the lift is therefore a function of the angle of attack. A variant of the above formulation of the problem that is useful in applications is to specify the lift of the airfoil instead of α. The angle of attack necessary to produce this lift is then determined by imposing the Kutta condition.

3. The Inverse Problem

In this section we describe the inverse, or design, problem of calculating the shape that a profile must have in order to achieve a given pressure distribution.

Suppose there is a flow past a profile such as the one depicted in Figure 2. If the velocity of the flow along the profile is written in the form

$$u(s) - iv(s) = Q(s)\, e^{-i\theta(s)} ,$$

then the direct problem consists of specifying the angle $\theta(s)$, which determines the shape of the airfoil, and solving for the flow. For the inverse problem, the values of $Q(s)$ are given and both the flow and the body are to be determined. Since Bernoulli's law (2.7) provides a correspondence between the values of q^2 and $c^2 = \text{constant}\cdot p^{(\gamma-1)/\gamma}$, we may formulate the inverse problem in terms of either q or ρ, and the choice of q is only a matter of mathematical convenience.

The inverse problem is seen to be a free boundary problem and is for this reason more complicated than the direct problem. The coordinates $(x(s),y(s))$ of the airfoil are now unknown and are to be determined from the knowledge of the velocity distribution $Q(s)$. We may write the relationship between ϕ and $Q(s)$ as an additional boundary condition

$$(2.15) \qquad \frac{d}{ds}\,\phi\big(x(s),y(s)\big) = Q(s)$$

on the interval $0 \leq s \leq \ell$, where ℓ is to be the total length of the airfoil.

It is also necessary to specify the constant in Bernoulli's law. In the direct problem the Mach number and speed at infinity are prescribed as in (2.14); for the design problem we give instead the value of the critical speed c_* in (2.7). This means that the Mach numbers of the flow along the airfoil are specified. The choice of c_* determines the type of the equation (2.8). If $c_* > \max |Q(s)|$, the flow will be subsonic and (2.9) will be elliptic; if there are points with $|Q(s)| > c_*$, the flow will be transonic and (2.9) will have mixed type.

We remark that with this formulation q_∞ is not specified as data, but must be determined along with ϕ and $\big(x(s),y(s)\big)$.

The fact that Q(s) is to be the velocity distribution of a flow past an airfoil places restrictions on the form Q(s) may have. A typical choice for Q(s) is illustrated in Figure 3. Q(s) must have a zero corresponding to the stagnation point that forms at the nose of the airfoil, and then must be nonzero along the surface of the airfoil until the tail is reached. At the tail the velocity must be continuous and may be taken to be nonzero provided the included angle ε at the trailing edge is zero.

$Q(s)$ must satisfy further compatibility conditions in order to determine profiles defined by simple closed curves.

Note that since the circulation of the flow is given by the integral of the speed along the profile, the lift of the airfoil can be calculated from the prescribed velocity distribution. The angle of attack may still be specified in the asymptotic form (2.13), since in the design problem the airfoil is free to rotate with respect to the fixed coordinate system so as to satisfy (2.12).

To summarize, the design problem is posed by specifying the speed distribution $Q(s)$, the critical speed c_*, and the angle of attack α. The potential of the flow and the airfoil coordinates are then obtained by solving the equations

$$(2.9)\qquad (c^2-\phi_x^2)\phi_{xx} - 2\phi_x\phi_y\phi_{xy} + (c^2+\phi_y{}^2)\phi_{yy} = 0 ,$$

$$(2.7)\qquad \frac{\phi_x^2+\phi_y^2}{2} + \frac{c^2}{\gamma-1} = \frac{1}{2}\,\frac{\gamma+1}{\gamma-1}\,c_*^2 ;$$

$$(2.10)\qquad \frac{\partial\phi}{\partial\nu}\big(x(s),y(s)\big) = 0 ;$$

$$(2.15)\qquad \frac{d}{ds}\,\phi\big(x(s),y(s)\big) = Q(s) ;$$

$$(2.12)\qquad \big|\underset{\sim}{u}\big(x(0),y(0)\big)\big| < \infty \quad ;$$

$$(2.13)\qquad \phi \sim q_\infty r\cos(\theta-\alpha) + \frac{\Gamma}{2\pi}\tan^{-1}\big(\beta\tan(\theta-\alpha)\big) .$$

III. DISCUSSION OF THE COMPUTATIONAL PROCEDURE

In this chapter we describe the method used to solve the design problem outlined in Section 2.3. In Section 3.4 we also provide a brief summary of the equations used to compute the boundary layer correction.

1. Overview of the Computation

The procedure we describe here is based on the Bauer, Garabedian, and Korn analysis code H [2,3,4] which solves the direct problem described in Section 2.2. The analysis routine computes the inviscid flow past a given airfoil in two steps. The region exterior to the airfoil in the z-plane is mapped conformally onto the interior of the unit circle in the ζ-plane, and the partial differential equation (2.9) for $\phi(x,y)$ is expressed in terms of the variables (r,ω), where $\zeta = r\,e^{i\omega}$. The resulting nonlinear equation is then approximated by a finite difference scheme which is solved by a relaxation procedure to provide the solution $\phi(r,\omega)$.

For the inverse problem, we are given the velocity distribution $Q(s)$ rather than the coordinates of the airfoil. The basic idea is to use $Q(s)$ to determine both

the mapping $z = f(\zeta)$ of the unit circle onto the desired airfoil and the potential ϕ. With this free boundary approach, the flow calculations are done in a fixed computational domain and the geometry is determined by introducing the appropriate mapping as an additional unknown. This results in coupled nonlinear equations for ϕ and for the mapping function f which we solve iteratively using existing routines in the analysis code.

The iterations go roughly as follows. We start with a first guess for the potential function which we take to be the incompressible solution $\phi^{(0)}$ obtained by replacing the partial differential equation (2.9) by Laplace's equation. The values of $\phi^{(0)}$ are used in the equation for the mapping function, which we solve for the approximation $z = f^{(1)}(s)$. Using the mapping $f^{(1)}$ in the flow equation then provides a better approximation $\phi^{(1)}(r,\omega)$ to the potential, and the process is repeated until the approximations converge. In terms of the analysis code, at each cycle the design mode starts with an approximation to the desired airfoil $P^{(n)}$, maps it to the unit circle, and solves for the flow past $P^{(n)}$ in the usual way. The resulting speed distribution is then compared to the desired speed distribution $Q(s)$, and a correction to $P^{(n)}$ is made to obtain the new approximation $P^{(n+1)}$. The iterations continue until the computed speed distribution agrees with the prescribed distribution $Q(s)$ within an acceptable accuracy.

We now describe this procedure in more detail. Consider a conformal mapping $z = f(\zeta)$ from the unit circle onto the exterior of an airfoil such as appears in Figure 2. We assume f has a pole at the origin and takes the point $\zeta = 1$ into the tail of the profile $(x(0), y(0))$. The included angle at the trailing edge will be taken to be zero, although the less important case $\varepsilon > 0$ could be treated similarly. The derivative of the map function has an expansion of the form

$$(3.1) \qquad \frac{dz}{d\zeta} = f'(\zeta) = -\frac{1-\zeta}{\zeta^2} \exp \sum_{k=0}^{\infty} c_k \zeta^k ,$$

where the behavior of the mapping at the trailing edge of the airfoil is taken into account by the factor $(1 - \zeta)$.

The mapping determines a boundary correspondence between the unit circle $\zeta = e^{i\omega}$ and the airfoil which we write as $s = s(\omega)$, where s is arc length along the profile. If we denote the inclination of the tangent to the airfoil by $\Theta(s)$, as in Figure 2, then on the unit circle $\zeta = e^{i\omega}$ we have

$$(3.2) \qquad f'(\zeta) = \left|\frac{dz}{d\zeta}\right| \exp\left\{i \arg \frac{dz}{d\zeta}\right\}$$
$$= \frac{ds}{d\omega} \exp\left\{i\left(\Theta\left(s(\omega)\right) - \omega - \frac{\pi}{2}\right)\right\}.$$

If $c_k = a_k + ib_k$, then (3.1) gives

$$(3.3a)\quad \log\left[\frac{1}{2\sin\frac{\omega}{2}}\frac{ds}{d\omega}\right] = \sum_{k=0}^{\infty} a_k \cos k\omega - b_k \sin k\omega$$

$$(3.3b)\quad \Theta\big(s(\omega)\big) + \frac{\omega}{2} + \pi = \sum_{k=0}^{\infty} b_k \cos k\omega + a_k \sin k\omega \ .$$

These equations show that the mapping is essentially determined once the correspondence $s = s(\omega)$ is known.

Under the change of variables $z = f(\zeta)$, $\zeta = r\, e^{i\omega}$, the partial differential equation (2.9) for ϕ becomes

$$(3.4)\quad r^2(c^2 - u^2)\tilde{\phi}_{rr} - 2ruv\tilde{\phi}_{r\omega} + (c^2 - v^2)\tilde{\phi}_{\omega\omega}$$

$$+ r(c^2 - v^2)\tilde{\phi}_r + \frac{1}{r}(u^2 + v^2)[r^3 uh_r + r^2 vh_\omega] = 0 \ ,$$

where $\tilde{\phi}(r,\omega) = \phi(x,y)$, $h^2 = |dz/d\zeta|^2$, $u^2 = \tilde{\phi}_r^2/h^2$, $v^2 = \tilde{\phi}_\omega^2/(r^2h^2)$, and c^2 is given by Bernouli's law (2.7). Note that the mapping function f appears in (3.4) through the Jacobian h. Furthermore, for the inverse problem the solution $\tilde{\phi}(r,\omega)$ of (3.4) can be used together with the data $Q(s)$ to provide boundary values for the determination of $f'(\zeta)$. The relation $\tilde{\phi}(1,\omega) = \phi\big(x(s(\omega)),y(s(\omega))\big)$ yields upon differentiation

$$(3.5)\qquad \frac{ds}{d\omega} = \frac{1}{Q(s(\omega))}\frac{\partial\tilde{\phi}}{\partial\omega}(1,\omega) \quad ,$$

which can be used in the expression (3.3a) for $\log|f'(e^{i\omega})|$.

The problem is therefore described by the equations (3.1), (3.3a), (3.5), and (3.4), together with the appropriate

boundary conditions for $\tilde{\phi}$ to supplement (3.4). The iterations used in the computation to solve this problem start with the approximation $\tilde{\phi}^{(0)}(r,\omega)$ provided by incompressible theory. If we introduce harmonic function $G(\zeta) = \log |\zeta^2(1 - \zeta)^{-1}f'(\zeta)|$, the iteration scheme then proceeds by solving in succession

$$\text{(3.6)} \qquad \frac{ds^{(n)}}{d\omega} = \frac{1}{Q\left(s^{(n)}(\omega)\right)} \frac{\partial \tilde{\phi}^{(n-1)}}{\partial \omega}(1,\omega) ,$$

$$\text{(3.7)} \quad \begin{cases} \Delta G^{(n)}(\zeta) = 0 \\ G^{(n)}(e^{i\omega}) = \log \left(\dfrac{1}{2\sin\frac{\omega}{2}} \dfrac{ds^{(n)}}{d\omega}(\omega)\right) \end{cases}$$

$$\text{(3.8)} \qquad h^{(n)}(\zeta) = \frac{|1 - \zeta|}{|\zeta|^2} \exp G^{(n)}(\zeta) ,$$

$$\text{(3.9)} \quad r^2(c_n^2-u_n^2)\tilde{\phi}_{rr}^{(n)} + ru_nv_n\tilde{\phi}_{r\omega}^{(n)} + (c_n^2-v_n^2)\tilde{\phi}_{\omega\omega}^{(n)}$$
$$+ r(c_n^2- v_n^2)\tilde{\phi}_r^{(n)} + \frac{1}{r}(u_n^2+v_n^2)[r^3u_nh_r^{(n)} + r^2v_nh_\omega^{(n)}] = 0$$

for $n = 1,2,3,\ldots$, where $u_n^2 = {\tilde{\phi}_r^{(n)}}^2/{h^{(n)}}^2$, $v_n^2 = {\tilde{\phi}^{(n)}}^2/(r^2{h^{(n)}}^2)$, and c_n^2 is given in terms of u_n^2 and v_n^2 by Bernoulli's law (2.7). The boundary conditions used to solve (3.9) are those of the direct problem.

The mapping computation (3.7) can be done rapidly using the fast Fourier transform to evaluate the coefficients appearing in a truncated series expansion for $G^{(n)}$.

The flow computation is performed using a nonconservative difference scheme similar to the one first developed by Murman and Cole [27]. Its main feature is type-dependent differencing which captures shocks over two mesh widths by effectively producing an artificial viscosity in the supersonic regions.

The iterative procedure works very well for subsonic flows, presumably because the initial guess is a good approximation to the solution. In fact we prove in Section 5.2 that a similar iteration converges to a solution provided the maximum Mach number in the flow is small enough.

For the case of transonic flow, the problem is complicated by the possible presence of shocks in the flows past the various approximations to the desired airfoil. A large gradient in the derivative of $\tilde{\phi}^{(n-1)}(1,\omega)$ appearing in (3.6) is undesirable, since a discontinuity in (3.3a) causes a logarithmic singularity in (3.3b), which is inconsistent with the assumed smoothness of the airfoil.

One way to avoid this difficulty is to solve the equations on a coarse mesh. The coefficient of the artificial viscosity implicit in the Murman-Cole scheme is of the order of a mesh width. If the grid is coarse enough, weak shocks are suppressed by this viscosity, as illustrated in Figure 4. This smoothing effect allows the process to converge even in the case of transonic flow. If the grid

is refined, unwanted shocks may appear in the flow, causing the iterations to diverge. On the other hand, a solution computed on too coarse a mesh may not accurately model the actual flow past the airfoil because of the smoothing effect of the artificial viscosity.

In order to operate on a fine mesh, we have added an additional smearing term to inhibit the formation of shocks. We describe this term in more detail in the next section. It has the form of an artificial viscosity multiplied by a coefficient $\varepsilon_1 \Delta\omega$, where $\Delta\omega$ is the mesh width in the angular direction. The factor ε_1 can be varied to change the amount of smoothing used. This permits the use of more viscosity in the early iterations when it is important to suppress shocks, and less viscosity towards the end of the computation when a more accurate solution is desired. The additional smoothing term therefore significantly increases the versatility of the design routine.

2. The Flow Computation

In this section we discuss the difference scheme used in the flow calculation and also give the form of the additional artificial viscosity term used in the design procedure.

For computational purposes it is convenient to remove the singularities of $\tilde{\phi}(r,\omega)$ and $h(r,\omega)$ by defining

$$\tilde{\phi}(r,\omega) = \frac{q_\infty e^{a_0}}{r} \cos\left(\omega + \alpha - b_0\right) + \Phi(r,\omega) , \quad h(r,\omega) = \frac{H(r,\omega)}{r^2} .$$

The equations for $\Phi(r,\omega)$ then become

$$(3.10) \quad r^2(c^2 - u^2)\Phi_{rr} - 2ruv\Phi_{r\omega} + (c^2-v^2)\Phi_{\omega\omega} + r(c^2-2u^2-v^2)\Phi_r$$
$$+ \frac{1}{r}(u^2 + v^2)[ruH_r + vH_\omega] = 0 ,$$

where

$$u = [r^2\Phi_r - q_\infty e^{a_0}\cos(\omega+\alpha-b_0)]/H ,$$

$$v = [r\Phi_\omega - q_\infty e^{a_0}\sin(\omega + \alpha - b_0)]/H ,$$

and c^2 is given by Bernoulli's law (2.7). The boundary conditions (2.13), (2.10), and (2.12) become

$$(3.11) \quad \Phi(0,\omega) = -\frac{\Gamma}{2\pi}\tan^{-1}\left(\beta \tan(\omega+\alpha-b_0)\right) ,$$

$$(3.12) \quad \Phi_r(1,\omega) = q_\infty e^{a_0}\cos(\omega + \alpha - b_0) ,$$

$$(3.13) \quad \Phi_\omega(1,0) = -q_\infty e^{a_0}\sin(\alpha - b_0) .$$

In the flow computation, centered differences are used to approximate the coefficients of Φ_{rr} , $\Phi_{r\omega}$, and $\Phi_{\omega\omega}$, as well as all of the lower order terms in (3.10). A rotated difference scheme due to Jameson [20] is used to evaluate the second derivatives. This method uses centered differences at all subsonic points. At supersonic points, one-sided differences that are retarded in the local stream direction are used to produce an artificial viscosity similar to (3.15) below. The resulting nonlinear algebraic equations are solved using line relaxation in the direction of the flow.

For the two-dimensional flows past a wing section that we consider, the direction of supersonic flow is to a good approximation alligned with the ω-coordinate direction. The original Murman-Cole scheme would suggest the approximation

$$(3.14) \qquad \Phi_{\omega\omega}(i\ \Delta r,\ j\ \Delta\omega) \approx \frac{\Phi_{ij}-2\Phi_{i,j-1}+\Phi_{i,j-2}}{(\Delta\omega)^2}$$

which is first order accurate at $(i\ \Delta r,\ j\ \Delta\omega)$. The dominant truncation error in (3.14) is $\Delta\omega\Phi_{\omega\omega\omega}(i\ \Delta r,\ j\ \Delta\omega)$, which has the effect of an artificial viscosity. We may consider this scheme as an approximation to the equation

$$(3.15) \quad r^2(c^2-u^2)\Phi_{rr}+\ \ldots = \Delta\omega \max\ [0,\ (v^2-c^2)]\Phi_{\omega\omega\omega}\ .$$

The artificial viscosity on the right is absent in the subsonic regions and the coefficient tends to zero at the sonic line.

In order to improve the convergence of the design routine, we have added an additional artificial viscosity to the flow equation (3.10). The added term has the form

$$\varepsilon_1 \Delta\omega \frac{\partial}{\partial\omega} [V(M) \ \phi_{\omega\omega}] \tag{3.16}$$

which is motivated by the Murman-Cole artificial viscosity appearing in (3.15). Here V(M) is a smooth function of the local Mach number M which vanishes for $M \leq M_0$ and is one for $M \geq M_1$. We choose the numbers M_0 and M_1 so that this term is effective across the sonic line. For example, we may use $M_0 = 0.85$ and $M_1 = 0.95$. This is in contrast to the behavior of the artificial viscosity in (3.15), which is switched off at the sonic line.

The term (3.16) is added directly to the rotated difference scheme, so that for $\varepsilon_1 = 0$ the original scheme remains unchanged. Figure 5 illustrates the smoothing effect of (3.16) for a flow with a weak shock on a fine mesh. Note that the shock does not appear on a cruder mesh.

By adding the term (3.16) to the partial differential equation (3.10) we can obtain satisfactory convergence of the design scheme on either crude or fine meshes as desired.

3. The Conformal Mapping

In this section we discuss the calculation of the mapping function from (3.3a), and we also explain how the Mach number M_∞, the coefficient of lift C_L, and the rotation factor b_0 are obtained.

During each design iteration we use the data $Q(s)$ and the previous estimate $\tilde{\phi}^{(n-1)}(r,\omega)$ for the potential function to calculate a boundary correspondence $s = s^{(n)}(\omega)$ between the unit circle and the nth approximation to the airfoil. The values $\tilde{\phi}^{(n-1)}(r,\omega)$ are obtained from the analysis routine, which uses dimensionless units that are normalized by the free stream velocity q_∞ and the chord length L of the profile. Since q_∞ and L are not specified for the design problem, it is necessary to adjust the scaling at each iteration to make the prescribed data compatible with the units used by the analysis routine.

In order to use the analysis routine we need to supply values for the free stream Mach number M_∞ and the lift coefficient C_L. To determine M_∞ we use the relation

$$(3.17) \qquad \frac{1}{2}\left(\frac{c_*^2}{q_\infty^2}\right)\frac{\gamma+1}{\gamma-1} = \frac{1}{2} + \frac{1}{(\gamma-1)M_\infty^2} ,$$

which results from Bernoulli's law (2.14) and (2.7). The speed q_∞ corresponding to the data $Q(s)$ and c_* is obtained iteratively. In the first cycle we use the value for $q_\infty^{(0)}$ provided by the incompressible solution. At the nth step,

$q_\infty^{(n)}$ is chosen to be the scale factor that minimizes the expression

$$(3.18) \qquad \sum_i \left[q_i^{(n-1)^2} - \frac{Q^2(s^{(n)}(\omega_i))}{q_\infty^{(n)^2}} \right]^2 ,$$

where the points ω_i are evenly distributed around the unit circle and $q_i^{(n-1)}$ are the velocities along the surface of the (n-1) st approximation to the airfoil as computed by the analysis routine.

The coefficient of lift supplied to the analysis routine,

$$C_L = \frac{\int_0^{\ell} Q(s')ds'}{\frac{1}{2}\, q_\infty L} = \frac{[\tilde{\phi}]}{\frac{1}{2}\, q_\infty L} ,$$

is also affected by the scaling and must be similarly adjusted.

In order to evaluate $s^{(n)}(\omega)$ we use the data $Q(s)$ and $\phi^{(n-1)}(1,\omega)$ as follows. Formula (2.15) is integrated to provide the function

$$\Phi_1(s) = \int_0^s Q(s')\, ds' .$$

Φ_1 has a minimum at the stagnation point of Q, say s_0, and is monotonic on either side of s_0. We define a smooth function

$$(3.19) \qquad \Phi_2(s) = \begin{cases} -\sqrt{\Phi_1(s) - \Phi_1(s_0)} , & s < s_0 , \\ +\sqrt{\Phi_1(s) - \Phi_1(s_0)} , & s > s_0 , \end{cases}$$

which is monotonically increasing and can be inverted. Both the speed Q and the arc length s may then be considered functions of the modified potential Φ_2.

At the nth stage of the iteration, the potential $\tilde{\phi}^{(n-1)}(1,\omega)$ is modified as in (3.19). The result is scaled to have the same range as Φ_2 and inserted into the appropriate expressions for Q and s. This provides an expression $s = s^{(n)}(\omega)$ which can be differentiated and used directly in (3.3a) instead of using (3.5), which requires special treatment at the stagnation point.

The series

$$(3.3a) \quad \log \left\{ \frac{1}{2 \sin \frac{\omega}{2}} \frac{ds^{(n)}}{d\omega} \right\} = \sum_{k=0}^{\infty} a_k^{(n)} \cos k\omega - b_k^{(n)} \sin k\omega$$

is truncated at N terms and the coefficients $a_0^{(n)}$ and $a_k^{(n)} + ib_k^{(n)}$, $k = 1,\ldots,N-1$ are obtained using a fast Fourier transform. This procedure does not provide the coefficient $b_0^{(n)}$, which determines the orientation of the airfoil with respect to the coordinate axes. To find $b_0^{(n)}$, we appeal to the Kutta condition

$$(3.13) \qquad \Phi_\omega(1,0) = - q_\infty e^{a_0} \sin(\alpha - b_0) .$$

At the nth stage of the iteration, $b_0^{(n)}$ is determined so that (3.13) will be satisfied as the iterations converge.

In summary, at each iteration we rescale the data to update M_∞ and C_L, and determine the mapping coefficients

a_0 and $c_k = a_k + ib_k$, $k = 1,\ldots,N-1$, which are used as input to the analysis routine. The analysis routine then provides the potential $\tilde{\phi}(r,\omega)$ along with the necessary rotation factor b_0 to complete the cycle. The iterations continue until the velocity along the airfoil computed by the analysis code agrees well enough with the prescribed data $Q(s)$. The remaining boundary conditions are automatically satisfied, since at each iteration we use the analysis routine to solve for the flow past a given airfoil.

4. The Boundary Layer Correction

For the computations to be of practical value it is important to supplement the inviscid equations discussed thus far with equations describing the flow near the surface of the wing section where viscous effects cannot be disregarded. To do this, the inviscid theory is used to design a profile with a finite thickness between the upper and lower surfaces at the trailing edge. Next a boundary layer correction is computed on the basis of the inviscid pressure distribution. The displacement thickness of the boundary layer is then subtracted from the coordinates of the inviscid profile. Thus the end results of the computations are the actual coordinates of the airfoil, a viscous boundary layer next to the surface of the airfoil, and inviscid flow outside the streamline determined by the boundary layer.

The method of Nash and Macdonald [28] is used to compute the turbulent boundary layer correction. The momentum thickness θ^* and the displacement thickness δ^* are calculated from the von Karmen momentum equation

$$(3.20) \qquad \frac{d\theta^*}{ds} + (2 + H - M^2)\frac{dq}{ds}\frac{\theta^*}{q} = \frac{\tau}{\rho q^2}$$

where $H = \delta^*/\theta^*$ is the shape factor and τ is the skin friction. M^2 and q are functions of arc length s determined by the inviscid solution, and H and τ are given by

semi-empirical formulas. The ordinary differential equation (3.20) is integrated from transition points (x_R, y_R) that must be prescribed on the upper and lower surfaces of the airfoil, and a starting value for θ^* is obtained from the specified Reynolds number of the flow.

Separation of the boundary layer is predicted when the Nash-Macdonald parameter

$$C_{sep} = - \frac{\theta^*}{q} \frac{dq}{ds}$$

exceeds 0.004. It is important to choose the input speed distribution so that C_{sep} remains around 0.003 on the upper surface near the trailing edge. This is our version of a criterion due to Stratford for avoiding boundary layer separation [35].

When theoretically designed airfoils are evaluated in wind tunnel tests, it is sometimes found that the effects of the boundary layer cause losses in lift an^ ^ther discrepancies between theory and experiment. However, airfoils designed with such a Stratford pressure distribution using a similar inverse formulation [4] have been found to meet their design specifications in wind tunnel testing.

IV. COMPUTATIONAL RESULTS

In this chapter we present some results produced by the design mode of the analysis code. We include a description of pressure distributions that generate profiles with no significant drag creep according to the analysis code. Possible extensions of the main ideas to other problems in transonic flow are discussed in Section 4.2.

1. The Design Procedure

The inverse method of airfoil design uses as input the pressure distribution rather than the airfoil coordinates. In order to obtain airfoils with acceptable drag levels, an appropriate pressure distribution must be prescribed. In this section we discuss a method of using the design mode that produces wing sections with low wave drag as predicted by the analysis mode.

Our first example appears in Figures 6 and 7. Figure 3 shows the speed distribution used to produce the airfoil of Figure 6. On the upper surface the speed distribution rises from the stagnation point at the nose to a flat section of supersonic values along the first sixty percent

of chord, and then falls into a Stratford distribution near the tail. The distribution over the lower surface is entirely subsonic and is arranged so that the lift is evenly distributed along the section, with aft loading at the tail. The profile has been provided with a gap at trailing edge so that a boundary layer correction can be removed from the displayed coordinates, as shown in Figure 7.

Our experience with the design routine to date indicates that drag creep can be avoided by designing the airfoil to have a small enough supersonic zone. The supersonic zone can be increased or decreased in size as desired by varying the critical speed c_* used in the design routine. If too large a supersonic zone is used at design, the middle part of the zone tends to collapse at speeds below design, giving rise to one or two shocks that can cause significant wave drag at off-design conditions. This effect is reduced by designing at a lower Mach number with a smaller supersonic zone.

Figure 8 shows an airfoil design with a speed distribution similar to that of Figure 6 but with the critical speed c_* lowered so that the supersonic zone is significantly larger. The pressure distribution was altered slightly near the nose and tail of the airfoil to retain the same thickness-to-chord ratio and about the same gap at the tail.

When drag rise curves are computed for these two airfoils, the profile designed with the larger supersonic zone exhibits drag creep as illustrated in Figure 9. The observed difference in the drag levels for the two profiles is due to increases in both the wave drag and the form drag of the second airfoil, although both airfoils have virtually identical form drag at subcritical speeds.

The design mode can also be used to improve the performance of airfoils by altering off-design pressure distributions. For example, we may exploit the fact that some shockless airfoils designed by hodograph procedures exhibit characteristic off-design distributions when evaluated near the design angle of attack with a lower Mach number (cf. [2], p. 96; [3], p. 131, p.143). The speed along much of the upper surface is roughly sonic, with a pronounced peak near the nose of the profile. Such peaky distributions also recall the experimental work of Pearcey [31], as well as Boerstal and Uijlenhoet [9] and Nieuland and Spee [29], who have published examples of shockless airfoils designed with peaky distributions.

We illustrate the use of this observation with another example. We begin with an airfoil that was obtained using the design mode with a Mach number $M_\infty = 0.745$. We use the analysis routine to compute flows past this profile with the same angle of attack but with smaller Mach numbers.

At $M_\infty = 0.710$ there results the distribution shown in Figure 10, with an upper surface distribution that resembles the characterisiic distributions except for a bump in the distribution around sixty percent of chord. This distribution is obtained as output from the code in the form of punched cards. We modify the distribution by removing the bump so that the distribution remains relatively flat along this section of the airfoil. The resulting distribution is used in the design mode to obtain the airfoil shown in Figure 11. In Figure 12 we display drag rise curves for this airfoil and a shockless airfoil with similar specifications designed by Dr. Jose Sanz using the hodograph code of [4]. The airfoil produced by the design mode of the analysis code compares quite favorably with the shockless airfoil. Figure 13 shows that a near-shockless flow is obtained at $M_\infty = 0.740$.

The two previous examples illustrate the procedure we use to obtain airfoils with low wave drag. We start with an upper surface speed distribution similar to the one appearing in Figure 3. This portion of the distribution determines the wave drag experienced by the airfoil at off-design conditions and also determines the growth of the boundary layer near the tail. To obtain airfoils with a given gap at the tail, thickness-to-chord ratio, and lift, the lower surface distribution should be modified as we

will indicate in Section 6.1. The value of c_* used determines the free stream Mach number M_∞ , as well as the size of the supersonic zone. In order to find the proper size for the supersonic zone, it may be necessary to make a tentative choice for c_* and use the analysis code to calculate a drag rise curve for the resulting profile. The size of the supersonic zone should be decreased if the wave drag is too high at speeds below design.

The size of the supersonic zone used at design determines the height of the pressure peak near the nose of the profile at off-design conditions, which in turn is related to the amount of wave drag occurring below design. The amount of wave drag near the design condition is effected by the curvature of the profile at the rear of the supersonic zone, which is governed by both the prescribed pressure distribution and the amount of artificial viscosity used in the design routine. Rather than attempting to adjust this area of the profile when it is in the most sensitive region of flow, we have found it more convenient to make any necessary modifications in an additional design run at a lower Mach number corresponding to the characteristic off-design condition as in Figures 8 and 9. To do this, the analysis mode is used to obtain the flow past the profile at the design angle of attack but with lower Mach numbers. At some Mach number

the flow should have a pressure peak near the nose followed by a section of nearly constant sonic flow. The pressure distribution over the region corresponding to the rear of the supersonic zone at the design condition is examined for any irregularities, and, if necessary, the pressure distribution is modified near this point so that it more closely resembles the characteristic off-design distribution of shockless airfoils.

In taking this approach, our philosophy is therefore to use a relatively simple upper surface distribution as in Figure 3 at the design condition. At this stage we adjust the remainder of the input distriubtion so that the airfoil has the desired specifications and we determine the size of the supersonic zone so that the off-design performance is acceptable. In doing so we operate on the fine mesh of code H with enough added artificial viscosity to ensure convergence of the scheme.

If the analysis mode is used to evaluate the airfoil at the design condition, the resulting pressure distribution usually agrees with the assigned pressure distribution except near the rear of the supersonic zone where the extra artificial viscosity used in the design mode has its largest effect. Rather than attempting to achieve better agreement between design and analysis in this region by using less artificial viscosity in the design mode, we instead go to the off-design condition to make any necessary

modifications to the profile at this point. Small corrections usually do not significantly alter the specifications of the airfoil that were determined at design. The result is a smooth profile with low wave drag at off-design conditions.

We discuss the implementation of this procedure, together with the necessary boundary layer correction, in Sections 6.1 and 6.2.

Figure 14 shows an airfoil with a larger supersonic zone designed on a fine mesh using a relatively small coefficient $\varepsilon_1 = 0.05$ in the additional artificial viscosity term (3.16). The appearance of the sonic line suggests the presence of a shock in the interior of the flow region which weakens as it approaches the profile. This picture illustrates the fact that this approach does not produce shockless airfoils, but can provide some control over the shock strength at the body by fitting a smooth pressure distribution.

2. Extensions of the Technique

A procedure similar to the one presented here would allow the design of transonic cascades with low wave drag. Codes which compute transonic flow past turbines and compressors by using relaxation schemes similar to the one in the analysis code used here have been written [19] and would presumably lend themselves to a similar design modification. An attractive feature of this approach would be avoiding the complicated paths of integration necessary for the design of transonic cascades using complex characteristics in the hodograph plane [4]. For example, it might prove possible to obtain cascades with a smaller gap-to-chord ratio than can be obtained using the hodograph method.

An important extension of this method is to the case of three-dimensional transonic flow past wing-body combinations. The results obtained so far with the design routine suggest that by choosing the proper pressure distribution, a satisfactory wing might be obtained with the mesh widths usually available to three-dimensional codes. Analysis codes that compute the transonic flow past a given wing are currently available [3,21]. It would be necessary to extend the method used in two dimensions to treat the more complicated free boundary. One possibility would be to use a separate conformal mapping at each wing section to define the wing.

V. A CONVERGENCE THEOREM

This chapter treats some theoretical aspects of the design problem. Section 1 describes the simplest design problem in incompressible flow, where the method is exact. Section 2 outlines a convergence proof for an iteration scheme similar to the one used in the computations. The estimates needed for the proof are described in Section 3.

1. The Incompressible Problem

In this section the design problem is illustrated for the elementary case of incompressible flow. The explicit solution obtained here is used in the next section as the basis for a convergence proof of an iteration scheme similar to the procedure outlined in Chapter 3 in the case of subsonic compressible flow.

To make the presentation as simple as possible we will consider the case of purely circulatory flow around a smooth object. The Kutta-Joukowski condition (2.12) is then unnecessary and the speed of the flow at infinity is zero, which simplifies the asymptotic representation (2.13). It is also convenient to formulate the problem in terms of the stream function ψ as well as the velocity potential ϕ.

We first establish some notation which will be useful

in the next section. Consider a flow circulating around a smooth body as indicated in Figure 15. We choose units so that the total arc length of the body is 2π, so that the density tends to one at infinity, and so that the maximum speed on the surface of the body is one. We measure arc length s from a fixed reference point on the body and express the coordinates as functions $(x(s),y(s))$ of s for $0 \leq s \leq 2$. The speed of the fluid along the body is denoted by

(5.1) $$Q(s) = |\underset{\sim}{u}(x(s),y(s))| \quad .$$

for $0 \leq s \leq 2\pi$. We let $\delta = \min Q(s) > 0$ so that $\delta \leq Q(s) \leq 1$. The potential function

(5.2) $$\Phi(s) = \int_0^s Q(s')\,ds'$$

is then monotonically increasing and $\Phi(2\pi)-\Phi(0) = -\Gamma \geq 2\pi\delta$, where $\Gamma < 0$ is the circulation of the flow.

It is convenient to consider $Q(s)$ as defined by (5.1) to be a 2π-periodic function defined for $-\infty < s < \infty$, and to consider Φ to be defined on the whole real line. The function

(5.3) $$s = S(\Phi)$$

inverse to $\Phi(s)$ then satisfies

$$S(\Phi + \Gamma) = S(\Phi) + 2\pi$$

for $-\infty < \Phi < \infty$, and the function

$$\hat{Q}(\Phi) = Q(S(\Phi)) \tag{5.4}$$

has period $-\Gamma$ over the real axis.

In this section the motion is assumed to be incompressible, which means we may take $\rho \equiv 1$ in (2.2). Formulas (2.8a) and (2.8b) then show that the complex function

$$\chi(z) = \phi(z) + i\psi(z) \tag{5.3}$$

is analytic with derivative

$$\frac{d\chi}{dz} = u - iv \ .$$

Near infinity, the asymptotic form analogous to (2.13) is

$$\chi(z) \sim \frac{\Gamma}{2\pi i} \log z \ . \tag{5.4}$$

We normalize χ so that

$$\psi(x(s),y(s)) = 0. \tag{5.5}$$

We begin by observing that the body is determined up to a rotation and translation by the function $Q(s)$. To see this, consider the conformal mapping $z = f(\zeta)$ which takes the interior of the unit circle in the ζ-plane onto the region exterior to the body in the z-plane. We assume the pole of f is located at $\zeta = 0$ and $f(1) = x(0) + iy(0)$.

In the ζ-plane, the complex potential χ assumes the simple form

$$\chi(\zeta) = \frac{-\Gamma}{2\pi i} \log \zeta,$$

and in particular for $\zeta = e^{i\omega}$,

$$\phi(e^{i\omega}) = -\frac{\Gamma\omega}{2\pi}$$

corresponds to the function $\Phi(s)$. This provides a correspondence $s = s(\omega)$ between the unit circle and the surface of the body of the form

$$(5.6) \qquad s = S\left(-\frac{\Gamma\omega}{2\pi}\right),$$

which is valid for all ω. To determine the body from the boundary correspondence (5.6), we note that if $F(\zeta) = -\zeta^2 f'(\zeta)$, we have

$$|F(e^{i\omega})| = \left| \frac{dz}{d\zeta}(e^{i\omega}) \right| = \left| \frac{ds}{d\omega}(\omega) \right|$$

and therefore

$$(5.7) \quad \log|F(e^{i\omega})| = \log\left|\frac{ds}{d\phi}\frac{d\phi}{d\omega}\right| = \log\left[\frac{-\Gamma}{2\pi\ \hat{Q}\left(\frac{-\Gamma\omega}{2\pi}\right)}\right]$$

This determines the boundary values of the harmonic function $\log|F(\zeta)|$. If $G(\zeta)$ is a conjugate harmonic function for $\log|F(\zeta)|$, we have

$$(5.8) \qquad f'(\zeta) = \zeta^{-2}|F(\zeta)| \exp i\{G(\zeta) + b_0\}$$

where b_0 is a real constant. The mapping $f(\zeta)$ is therefore determined up to a translation and a rotation by $Q(s)$. The flow

$$(5.9) \qquad u - iv = \frac{\Gamma}{2\pi i\zeta \; f'(\zeta)}$$

is determined as well up to a multiplicative factor e^{-ib_0}. This is all that can be expected, since a Euclidean transformation of the coordinates leaves the speed distribution Q unchanged.

We may now change our viewpoint and let the formulas (5.7), (5.8), and (5.9) determine a nonzero analytic function $f'(\zeta)$ and a flow $u - iv(\zeta)$, say with $b_0 = 0$, from a prescribed function $Q(s)$. Provided that the function $f(\zeta)$ determines a reasonable profile, the boundary condition (5.7) shows that the resulting flow $u-iv(z)$ does have magnitude Q on the body, since

$$|u-iv\big(x(s),y(s)\big)| = |\tfrac{d}{ds}\,\phi\big(x(s),y(s)\big)| = |\tfrac{\partial\phi}{\partial\omega}||\tfrac{d\omega}{ds}| = Q(s) \;.$$

The question arises whether every smooth, periodic function $Q(s)$ determines a reasonable profile $\big(x(s),y(s)\big)$. This is not the case. For example, if

$$f'(\zeta) = -\frac{1}{\zeta^2} \exp \sum_{n=0} c_n \zeta^n \;,$$

then

$$0 = \oint_{\text{body}} dz = -\oint_{|\zeta|=1} f'(\zeta)\, d\zeta = 2\pi i \; c_1 \; e^{c_0} \;,$$

where

$$c_1 = \frac{1}{\pi} \int_0^{2\pi} \log |F(\zeta)| \; e^{-i\omega} \, d\omega \; .$$

This imposes a compatibility condition

$$0 = \int_0^{2\pi} \log \; \hat{Q}\left(\frac{-\Gamma\omega}{2\pi}\right) e^{-i\omega} \, d\omega$$

on Q(s) in order to obtain a closed body in the z-plane: In the transonic design problem for flow past an airfoil, an analogous condition on Q allows one to design airfoils that have a finite thickness between the upper and lower surfaces of the trailing edge in order to represent a wake extending downstream from the tail of the airfoil.

A more subtle detail is that the mapping $z = f(\zeta)$ determined by Q(s) may define a profile with self-intersecting boundaries. In the airfoil design problem, this consideration is important since the body determined by Q(s) may have so much curvature that the top and bottom surfaces overlap.

Nevertheless, we emphasize that the formulas (5.7), (5.8), and (5.9) do provide a locally one-one mapping $z = f(\zeta)$ and a flow $u-iv(\zeta)$ if only Q(s) is positive and periodic. Similarly, the numerical computations for the transonic design problem are possible under very mild requirements on Q(s), and the process converges whether or not the resulting airfoil is physically realizable.

It falls to the user of our computer code to make the modifications of Q(s) necessary to obtain an acceptable geometry.

2. A Convergence Proof for the Compressible Case

We now consider the more complicated case of subsonic compressible flow around a smooth obstacle. We wish to show that an iterative procedure similar to the one used in the numerical computation converges to a solution of the inverse problem. The approach we take exploits the idea that incompressible flow can be considered to be a limiting case of compressible flow as the speed of sound c becomes infinitely large. The convergence proof requires the maximum Mach number in the flow to be small, which can be assured by taking the prescribed critical speed c_* large enough. In this case we obtain a Poisson problem with nonlinear inhomogeneous terms which we solve by iteration. We are able to use standard estimates expressed in terms of Hölder continuity to show that the iterative procedure defines a contraction, so that the iterations converge. Less restrictive results could probably be obtained using deeper techniques from the mathematical theory of subsonic flows. However, the proof outlined here is a satisfactory illustration of the computational procedure, which is our main concern.

For the subsonic design problem, the critical speed c_* is given in addition to the speed distribution $Q(s)$. We continue to use the conventions of Section 5.1, with units chosen so that ρ approaches one at infinity and $\max Q(s) = 1$. As in the computational procedure, we solve

the free boundary problem by finding both the map $z = f(\zeta)$ from the interior of the unit circle to the region exterior to the desired body and by solving for the compressible flow $\underset{\sim}{u}(\zeta)$. We show that for a fixed speed distribution $Q(s)$, if c_* is sufficiently large we may construct a map $z = f(\zeta)$ and a stream function $\psi(\zeta)$, both depending on c_*, which approach the corresponding incompressible solutions determined by $Q(s)$ as $c_* \to \infty$.

For compressible flow, the velocity potential ϕ and stream function ψ, considered as functions of the variable $\zeta = \xi + i\eta$, satisfy

(5.10a) $$\rho\phi_\xi = \psi_\eta ,$$

(5.10b) $$\rho\phi_\eta = -\psi_\xi .$$

We choose to work with ψ instead of ϕ so that we may use the Dirichlet boundary condition $\psi = 0$ rather than the Neumann condition $\partial\phi/\partial\nu = 0$.

By eliminating ϕ from (5.10a) and (5.10b) we obtain the equation

(5.11) $$\Delta\psi = \frac{1}{c^2}\left[u^2\psi_{\xi\xi} + 2uv\psi_{\xi\eta} + v^2\psi_{\eta\eta} + \frac{q^2}{2|f'|}\left(\psi_\xi |f'|_\xi + \psi_\eta |f'|_\eta\right)\right] ,$$

where $u^2 = \psi_\eta{}^2/\rho^2|f'|^2$, $v^2 = \psi_\xi{}^2/\rho^2|f'|^2$, $q^2 = u^2 + v^2$, and $c^2 = (\gamma+1)c_*^2\rho^{\gamma-1}/2$. The density ρ can be obtained from Bernoulli's law in the form

$$(5.12)\qquad \frac{\psi_\xi^2 + \psi_\eta^2}{2\rho^2|f'(\zeta)|^2} + \frac{1(\gamma+1)}{2(\gamma-1)} c_*^2\rho^{\gamma-1} = \frac{1}{2}\,\frac{\gamma+1}{\gamma-1}\; c_*^2 \ .$$

The solution we shall obtain has the asymptotic form

$$\psi \sim \frac{\Gamma}{2\pi} \log|\zeta|$$

as $\zeta \to 0$, where the circulation Γ is given by

$$\Gamma = -\int_0^{2\pi} Q(s)\, ds \ .$$

We have $2\pi \geq -\Gamma \geq 2\pi\delta$, with $\delta = \min Q(s) > 0$.

We again consider the analytic function $F(\zeta) = -\zeta^2 f'(\zeta)$, which satisfies the by now familiar boundary condition

$$(5.13)\qquad \log|F(e^{i\omega})| = \log\left|\frac{1}{\hat{Q}(\phi(e^{i\omega}))}\;\frac{\partial}{\partial\omega}\,\phi(e^{i\omega})\right| \ .$$

Using (5.10) we have $\rho\ \partial\phi/\partial\omega = -\partial\psi/\partial r$, so that we may express the potential function on the boundary in the form

$$(5.14)\qquad \phi(\omega) = -\int_0^{\omega} \frac{1}{\rho(e^{i\omega})}\,\frac{\partial\psi}{\partial r}\,(e^{i\omega})\; d\omega \ .$$

If we knew the solution $\psi(re^{i\omega})$, (5.14) could be used in the boundary condition (5.13) for the determination of $|f'(\zeta)|$.

We consider the expressions

$$(5.15)\qquad \Psi(\zeta) = \psi(\zeta) - (\Gamma/2\pi)\log|\zeta| \ ,$$

$$(5.16)\qquad H(\zeta) = \log|F(\zeta)| - \log|F_0(\zeta)| ,$$

where $-\zeta^{-2}F_0(\zeta)$ is the derivative of the conformal mapping obtained by solving the incompressible problem with the same data $Q(s)$, as described in Section 5.1. Ψ and H are perturbations of the corresponding incompressible solutions. Note that Ψ and H are not singular at the point $\zeta = 0$, in contrast to both the mapping $f(\zeta)$ and the stream function $\psi(\zeta)$.

Substitution of the expressions (5.13) and (5.14) into the equations (5.11) and (5.13) for ψ and F shows that Ψ and H must satisfy equations of the form

$$(5.17)\qquad \begin{cases} \Delta\Psi = M[\Psi,H,c_*] \\ \Psi(e^{i\omega}) = 0 \end{cases} ,$$

$$(5.18)\qquad \begin{cases} \Delta H = 0 \\ H(e^{i\omega}) = N[\Psi,H,c_*] \end{cases} ,$$

where M and N depend nonlinearly on Ψ and H and formally tend to zero as $c_*^2 \to \infty$. We give the explicit form of M and N in the next section. M is a function of Ψ and the partial derivatives of Ψ up to second order, H and $\log|\Gamma_0|$ and their first order derivatives, and the variables ξ and η. N involves the boundary values of H,Ψ , and the normal derivative $(\partial\Psi/\partial r)(e^{i\omega})$ in a manner similar to (5.13) and (5.14). The function $Q(s)$ enters the problem through the term $N[\Psi,H,c_*]$.

We are interested in solving equations (5.16) and (5.17) by iteration. We set $\Psi^{(0)} = H^{(0)} = 0$, and formally define $\Psi^{(n+1)}$ and $H^{(n+1)}$ as solutions of

$$(5.19) \qquad \begin{cases} \Delta\Psi^{(n+1)} = M[\Psi^{(n)}, H^{(n)}, c_*] \\ \Psi^{(n+1)}(e^{i\omega}) = 0 \end{cases}$$

$$(5.20) \qquad \begin{cases} \Delta H^{(n+1)} = 0 \\ H^{(n+1)}(e^{i\omega}) = N[\Psi^{(n+1)}, H^{(n)}, c_*] \ . \end{cases}$$

We denote this operation by

$$(5.21) \qquad \left(\Psi^{(n+1)}, H^{(n+1)}\right) = L\left(\Psi^{(n)}, H^{(n)}, c_*\right).$$

We wish to show that for a given $Q(s)$, if c_* is large enough the iterates satisfy an inequality of the form

$$(5.22) \quad \|\Psi^{(n+1)}-\Psi^{(n)}\| + \|H^{(n+1)}-H^{(n)}\| \le \theta\left(\|\Psi^{(n)}-\Psi^{(n-1)}\| + \|H^{(n)}-H^{(n-1)}\|\right)$$

with $\theta < 1$, which implies convergence of the iteration. We therefore must examine how the solutions $\Psi^{(n+1)}$ and $H^{(n+1)}$ of (5.19) and (5.20) depend on the functions $\Psi^{(n)}$ and $H^{(n)}$, and we must define the norms to be used in (5.22). It turns out that since M and N are formally small as $c_* \to \infty$, we may succeed by using basic estimates for Laplace's equation which are expressed in terms of Hölder continuity.

Let D denote the closed unit disk and consider the space $C_{n+\alpha}(D)$ consisting of n-times continuously differentiable functions u defined in D with finite norm

$$\|u\|_{n+\alpha} = \sup_{\substack{\zeta\in D\\ i,j\leq n}} \left|\partial_\xi^i \partial_\eta^j\, u(\zeta)\right| +$$

$$+ \sup_{\substack{\zeta_1,\zeta_2\in D\\ i+j=\eta}} \frac{|\partial_\xi^i\partial_\eta^j\, u(\zeta_1) - \partial_\xi^i\partial_\eta^j\, u(\zeta_2)|}{|\zeta_1-\zeta_2|^\alpha}$$

where n is a nonnegative integer and $0 < \alpha < 1$. $C_{n+\alpha}(D)$ is a complete space with this norm. In addition, if $u \in C_{n+\alpha}(D)$, then $u \in C_{n'+\alpha'}(D)$ when $n' \leq n$ and $\alpha' \leq \alpha$, and $\|u\|_{n'+\alpha'} \leq \|u\|_{n+\alpha}$. If $u_1, u_2 \in C_{n+\alpha}(D)$, then $u_1 \cdot u_2 \in C_{n+\alpha}(D)$ and $\|u_1\cdot u_2\|_{n+\alpha} \leq K_{n+\alpha}\|u_1\|_{n+\alpha}\|u_2\|_{n+\alpha}$. If $u \in C_{n+\alpha}(D)$ and G is (n+1)-times continuously differentiable on the real axis, then $G(u(\zeta)) \in C_{n+\alpha}(D)$.

We also need the idea of Hölder continuous boundary values. If $g(\omega)$ is an n-times continuously differentiable, 2π-periodic function defined for all ω, we will say $g \in C_{n+\alpha}(\partial D)$ if the norm

$$\|g\|_{n+\alpha,\partial D} = \sup_{\substack{\omega\\ i\leq n}} |\partial_\omega^i g(\omega)| + \sup_{\omega_1,\omega_2} \frac{|\partial_\omega^n g(\omega_1) - \partial_\omega^n g(\omega_2)|}{|\omega_1-\omega_2|^\alpha}$$

is finite. The norm $\|\cdot\|_{n+\alpha,\partial D}$ has properties similar to $\|\cdot\|_{n+\alpha}$. Note that if $f(\zeta) \in C_{n+\alpha}(D)$, then $f(e^{i\omega}) \in C_{n+\alpha}(\partial D)$ and $\|f(e^{i\omega})\|_{n+\alpha,\partial D} \leq K'_{n+\alpha}\|f\|_{n+\alpha}$.

For any fixed α, $0 < \alpha < 1$, we will show $\Psi^{(n)} \in C_{2+\alpha}(D)$ and $H^{(n)} \in C_{1+\alpha}(D)$ by using the following fact, which is an elementary instance of the more general a priori esti-estimates of Schauder [6]:

THEOREM. Let $u(\zeta)$ be the solution of

$$\begin{cases} \Delta u = f \\ u(e^{i\omega}) = \phi \end{cases}$$

where $f \in C_\alpha(D)$ and $\phi \in C_{2+\alpha}(\partial D)$. Then $u \in C_{2+\alpha}(D)$ and

$$\|u\|_{2+\alpha} \le K_3\left(\|f\|_\alpha + \|\phi\|_{2+\alpha,\partial D}\right) , \tag{5.23}$$

where K_3 depends only on α.

We will also use the following consequence of (5.23).

COROLLARY. Let $\tilde{u}(\zeta)$ be the solution of

$$\begin{cases} \Delta \tilde{u} = 0 \\ \tilde{u}(e^{i\omega}) = \tilde{\phi} \end{cases}$$

where $\tilde{\phi} \in C_{1+\alpha}(\partial D)$. Then $\tilde{u} \in C_{1+\alpha}(D)$ and

$$\|\tilde{u}\|_{1+\alpha} \le K_4\|\tilde{\phi}\|_{1+\alpha,\partial D} , \tag{5.24}$$

where K_4 depends only on α.

This result can be obtained from the previous theorem (5.23) by setting

$$\tilde{u}(re^{i\omega}) = \frac{\partial}{\partial\omega}\left(u(re^{i\omega})\right) + \left(\frac{1}{2\pi}\right)\int_0^{2\pi} \tilde{\phi}(\omega')\, d\omega' ,$$

where u is obtained by solving

$$\begin{cases} \Delta u = 0 \\ u(e^{i\omega}) = \int_0^\omega \tilde{\phi}(\omega')\, d\omega' - \frac{\omega}{2\pi} \int_0^{2\pi} \tilde{\phi}(\omega')\, d\omega' . \end{cases}$$

We assume the data $Q(s)$ is three times continuously differentiable. The expressions (5.2), (5.3), and (5.4) show that the function $\hat{Q}(\Phi)$ is also that smooth, and we set

$$K_Q = \sup_{\substack{-\infty<\Phi<\infty \\ j=1,2,3}} \left| \frac{d^j}{d\Phi^j} \hat{Q}(\Phi) \right| \tag{5.25}$$

Using (5.24) we see that the harmonic function $\log|F_0(\zeta)|$ determined by (5.7) is in $C_{1+\alpha}(D)$ with $\|\log|F_0(\zeta)|\|_{1+\alpha} \leq K_5'$, where K_5' depends only on K_Q, δ, and α. This implies $|F_0(\zeta)|^2 \in C_{1+\alpha}(D)$ with a similar bound $\||F_0(\zeta)|^2\|_{1+\alpha} \leq K_5$.

Consider the closed subset $B_{(r_1,r_2)}$ of $C_{2+\alpha}(D)\times C_{1+\alpha}(D)$ defined as

$$B_{(r_1,r_2)} = \left\{(\Psi,H) : \|\Psi\|_{2+\alpha} \leq r_1,\ \|H\|_{1+\alpha} \leq r_2\right\}$$

In order to show the iteration scheme (5.21) converges, we first establish the following

LEMMA A. There is a constant K_A depending only on α, γ, $\delta = \min Q(s)$, and K_Q such that for $c_* > K_A$, the

operator $L(\cdot,\cdot,c_*)$ formally defined by (5.19) and (5.20) is well defined on $B_{(\delta/2,1)}$ and maps $B_{(\delta/2,1)}$ into itself. Thus the iterates $\Psi^{(n+1)}$ and $H^{(n+1)}$ all exist and satisfy $\|\Psi^{(n+1)}\|_{2+\alpha} \leq \delta/2$, $\|H^{(n+1)}\|_{1+\alpha} \leq 1$.

We sketch the derivation of the estimates necessary for the proof of Lemma A in the next section. There we show that if $(\Psi_1,H_1) \in B_{(\delta/2,1)}$ and if c_* is large enough, we have $M[\Psi_1,H_1,c_*] \in C_\alpha(D)$ and $N[\Psi_1,H_1,c_*] \in C_{1+\alpha}(\partial D)$, with $\|M[\Psi_1,H_1,c_*]\|_\alpha = K_6/c_*^2$ and $\|N[\Psi_1,H_1,c_*]\|_{1+\alpha,\partial D} \leq K_6(1/c_*^2 + \|\Psi_1\|_{2+\alpha})$, where K_6 depends only on α,γ,δ, and K_Q. Using the basic estimates (5.23) and (5.24), we then have that the functions $(\Psi_2,H_2) = L(\Psi_1,H_1,c_*)$ defined by solving (5.15) and (5.16) satisfy

$$\|\Psi_2\|_{2+\alpha} \leq K_7/c_*^2 , \tag{5.26}$$

$$\|H_2\|_{1+\alpha} \leq K_7/c_*^2 , \tag{5.27}$$

with K_7 depending on α,γ,δ, and K_Q, which establishes Lemma A with $K_A^2 = K_7/\delta$.

Lemma A shows that the functions $\Psi^{(n)}$ and $H^{(n)}$ can indeed be generated for $n = 1,2,\ldots$. In order to show convergence, we have

LEMMA B. There is a constant K_B depending only on α,γ,δ, and K_Q such that for $c_* > K_B$, the operator $L(\cdot,\cdot,c_*)$ defined on $B_{(\delta/2,1)}$ is a contraction. More

specifically, if $(\Psi_2,H_2) = L(\Psi_1,H_1,c_*)$ and $(\Psi_4,H_4) = L(\Psi_3,H_3,c_*)$, we have

$$\|\Psi_2-\Psi_4\|_{2+\alpha}+\|H_2-H_4\|_{1+\alpha} \leq \theta\left(\|\Psi_1-\Psi_3\|_{2+\alpha}+\|H_1-H_3\|_{1+\alpha}\right)$$

where $\theta < 1$. Thus the iterates $\Psi^{(n)}$ and $H^{(n)}$ form Cauchy sequences in the complete spaces $C_{2+\alpha}(D)$ and $C_{1+\alpha}(D)$.

To establish Lemma B we consider the expressions

$$\text{(5.28)}\qquad \begin{aligned} &\Delta(\Psi_2-\Psi_4) = M[\Psi_1,H_1,c_*] - M[\Psi_3,H_3,c_*] \\ &\Psi_2(e^{i\omega}) - \Psi_4(e^{i\omega}) = 0 \end{aligned}$$

$$\text{(5.29)}\qquad \begin{aligned} &\Delta(H_2-H_4) = 0 \\ &H_2(e^{i\omega}) - H_4(e^{i\omega}) = N[\Psi_2,H_1,c_*] - N[\Psi_4,H_3,c_*] . \end{aligned}$$

In the next section we see that

$$\begin{aligned} &\|M[\Psi_1,H_1,c_*] - M[\Psi_3,H_3,c_*]\|_\alpha \\ &\qquad \leq (K_9/c_*^2)(\|\Psi_1-\Psi_3\|_{2+\alpha}+\|H_1-H_3\|_{1+\alpha}), \end{aligned}$$

$$\begin{aligned} &\|N[\Psi_2,H_1,c_*] - N[\Psi_4,H_3,c_*]\|_{1+\alpha,\partial D} \\ &\qquad \leq K_9\left(\|\Psi_2-\Psi_4\|_{2+\alpha}+\|H_1-H_3\|_{1+\alpha}/c_*^2\right) , \end{aligned}$$

where K_9 dpends on α,γ,δ, and K_Q. Applying the basic estimates (5.23) and (5.24) to equations (5.28) and (5.29),

we have

$$\|\Psi_2-\Psi_4\|_{2+\alpha} \le (K_{10}/c_*^2)\left(\|\Psi_1-\Psi_3\|_{2+\alpha}+\|H_1-H_3\|_{1+\alpha}\right) ,$$

$$\|H_2-H_4\|_{1+\alpha} \le K_{10}\left(\|\Psi_2-\Psi_4\|_{2+\alpha} + \|H_1-H_3\|_{1+\alpha}/c_*^2\right) ,$$

which together yield the statement of Lemma B with $K_B^2 = K_{10}(1 + 3K_{10})$ and $\theta = K_B^2/c_*^2$.

The Cauchy sequences of iterates $\Psi^{(n)}$ and $H^{(n)}$ therefore converge to functions $\Psi \in C_{2+\alpha}(D)$ and $H \in C_{1+\alpha}(D)$ which must satisfy $(\Psi,H) = L(\Psi,H,c_*)$ if $c_* \ge K_B$. We note that since L is a contraction, Ψ and H are the only solutions of (5.16) and (5.17) with $\|\Psi\|_{2+\alpha} \le \delta/2$ and $\|H\|_{1+\alpha} \le 1$. Moreover, the expressions (5.26) and (5.27) show that the solution satisfies

$$\|\Psi\|_{2+\alpha} + \|H\|_{1+\alpha} \le 2K_7/c_*^2 .$$

Recalling that Ψ and H are perturbation quantities representing the difference between the compressible and incompressible solutions for a given Q(s), we see that the compressible solution indeed tends to the corresponding incompressible solution as the critical speed $c_* \to \infty$. This fact, together with the explicit solution available in the incompressible case, is the underlying basis of the above convergence proof.

3. Inequalities for the Convergence Theorem

In this section we consider in more detail the inhomogeneous terms $M[\Psi,H,c_*]$ and $N[\Psi,H_1,c_*]$ appearing in equations (5.17) and (5.18) and indicate how the estimates mentioned in the discussions of Lemmas A and B are derived.

A calculation shows that the inhomogeneous term $M[\Psi,H,c_*]$ of (5.17) has the form

$$(5.28)\quad M[\Psi,H,c_*] = \frac{T(\xi_i,\Psi_{\xi_i},\Psi_{\xi_i\xi_j},H_{\xi_i},\log|F_0|_{\xi_i})}{c_*^2\rho^{\gamma+1}|\Gamma_0|^2 \exp 2H}$$

where $\xi_1 = \xi$, $\xi_2 = \eta$, and T is a third degree polynomial in its arguments with coefficients depending only on the circulation Γ. The term $N[\Psi,H,c_*]$, takes the form

$$(5.29)\quad N[\Psi,H,c_*] = \log\left[1 + \frac{2\pi}{\Gamma}\frac{\partial\Psi}{\partial r}(e^{i\omega})\right]$$

$$- \log \rho(e^{i\omega}) - \log[\hat{Q}(\Phi(\omega))/\hat{Q}(-\Gamma\omega/2\pi)]$$

Here the density ρ can be defined by Bernoulli's law (5.12) to be a function $\rho = R(\tilde{q}^2/c_*^2)$, where $\tilde{q}^2 = (\psi_\xi^2 + \psi_\eta^2)/|f'|^2$, valid for $0 \le \tilde{q}^2 \le 2K_1c_*^2$, with $R(0) = 1$. R is the subsonic branch of the multiple-valued function giving ρ in terms of the gradient of the stream function. The constant K_1 depends on γ, $2K_1 = [2/(\gamma+1)]^{2/(\gamma-1)}$. The only information about R that we need is that for

$\tilde{q}^2 \leq K_1 c_*^2$, there is a constant K_2 depending on γ such that $1 \geq R \geq K_2^{-1} > 0$ and $|R'|, |R''| \leq K_2$. Recalling the expressions (5.15) and (5.16) giving Ψ and H in terms of ψ and f', we have that

$$(5.30) \quad \tilde{q}^2 = \frac{1}{|F_0|^2 \exp 2H} \left[\left(\frac{\Gamma}{2\pi}\right)^2 |\zeta|^2 - \frac{\Gamma|\zeta|^2}{\pi} (\xi\Psi_\xi + \eta\Psi_\eta) + |\zeta|^4 (\Psi_\xi^2 + \Psi_\eta^2) \right].$$

The term $\Phi(\omega)$ appearing in the expression (5.28) for $N[\Psi, H, c_*]$ is written in the form

$$(5.31) \quad \Phi(\omega) = (\Gamma/c) \int_0^\omega \frac{1}{\rho(e^{i\omega})} \left[\frac{\Gamma}{2\pi} + \frac{\partial \Psi(e^{i\omega})}{\partial r} \right] d\omega$$

rather than (5.14), where the constant c is given by

$$(5.32) \qquad c = - \int_0^{2\pi} \frac{1}{\rho(e^{i\omega})} \left[\frac{\Gamma}{2\pi} + \frac{\partial \Psi(e^{i\omega})}{\partial r} \right] d\omega \; .$$

The factor c is introduced to make sure $\Phi(\omega+2\pi) - \Phi(\omega) = -\Gamma$, so that $\hat{Q}(\Phi(\omega))$ is 2π-periodic and smooth.

We first describe the inequalities used in the proof of Lemma A. Since the expressions (5.28), (5.29), (5.30), and (5.31) are rather complicated, we do not attempt to present all the details. We have chosen the data Q(s) to be smooth enough so that nothing more sophisticated than the mean value theorem is needed.

We assume $(\Psi,H) \in B_{(\delta/2,1)}$. We claim that if c_* is large enough, $M[\Psi,H,c_*] \in C_\alpha(D)$ with $\|M[\Psi,H,c_*]\|_\alpha \le K_6/{c_*}^2$, and $N[\Psi,H,c_*] \in C_{1+\alpha}(D)$ with $\|N[\Psi,H,c_*]\|_{1+\alpha,\partial D} \le$ $\le K_6\left(1/{c_*}^2 + \|\Psi\|_{2+\alpha}\right)$. In the following expressions, K_j will denote constants depending only on α,δ,γ, and K_Q.

Consideration of the expression (5.30) for $\tilde{q}^2$ shows that $\tilde{q}^2 \in C_{1+\alpha}(D)$ with $\|\tilde{q}^2\|_{1+\alpha} \le K_9'$. Recalling the properties of the function $\rho = R(\tilde{q}^2/c_*^2)$ defined by (5.12), we see that $\rho \in C_{1+\alpha}(D)$ with $\|\rho\|_{1+\alpha} \le K_{10}'$, provided that we have chosen $c_*^2 \ge K_9'/K_1$. This choice keeps $\tilde{q}^2/{c_*}^2$ on the subsonic branch of the density-stream function relation. We then see from (5.28) that $M[\Psi,H,c_*] \in C_\alpha(D)$ and we obtain $\|M[\Psi,H,c_*]\|_\alpha \le K_{11}'/{c_*}^2$.

We next observe that each of the three terms on the right hand side of the expression (5.29) for $N[\Psi,H,c_*]$ are in $C_{1+\alpha}(\partial D)$. The first term is well behaved since we have $|\Gamma| \ge 2\pi\delta$ and $\|\Psi\|_{2+\alpha} \le \delta/2$, giving $|(2\pi/\Gamma)(\partial\Psi/\partial r)(e^{i\omega})| \le \frac{1}{2}$, and its norm is bounded by a constant times $\|\Psi\|_{2+\alpha}$. The second term $\log \rho(e^{i\omega})$ is also well behaved since ρ is bounded away from zero, and we can estimate

$$\|\log \rho(e^{i\omega})\|_{1+\alpha,\partial D} \le K_{12}/{c_*}^2.$$

The desired factor $1/{c_*}^2$ is essentially due to the fact that

$$\log \rho(e^{i\omega}) = \log R(\tilde{q}^2(e^{i\omega})({c_*}^2) - \log R(0)$$

$$= \frac{\tilde{q}^2(e^{i\omega})}{{c_*}^2} \int_0^1 \frac{R'(t\tilde{q}^2/{c_*}^2)}{R(t\tilde{q}^2/{c_*}^2)}\,dt$$

by the mean value theorem.

The last term in (5.2a) can be estimated by

(5.31) $\|\log \hat{Q}(\Phi(\omega))-\log \hat{Q}(-\Gamma\omega/2\pi)\|_{1+\alpha,\partial D} \leq K_{13}(1/c_*^2 + \|\Psi\|_{2+\alpha}$.

This inequality is more complicated and we sketch it as follows. Using the fact that $\hat{Q}$ is three times continuously differentiable one can easily obtain

$$\|\log \hat{Q}(\Phi(\omega)) - \log \hat{Q}(-\Gamma\omega/2\pi)\|_{1+\alpha,\partial D}$$
$$\leq K_{14}\|\Phi(\omega) + \Gamma\omega/2\pi\|_{1+\alpha,\partial D} .$$

We then express $\Phi(\omega) + \Gamma\omega/2\pi$ in the form

$$\Phi(\omega) + \Gamma\omega/2\pi = \frac{\Gamma+c}{c}\int_0^\omega \frac{1}{\rho}\left[\frac{\Gamma}{2\pi} + \frac{\partial\Psi}{\partial r}\right] d\omega$$
$$- \int_0^\omega \frac{1-\rho}{\rho}\left[\frac{\Gamma}{2\pi} + \frac{\partial\Psi}{\partial r}\right] d\omega - \int_0^\omega \frac{\partial\Psi}{\partial r} d\omega$$

and estimate the three expressions on the right in terms of $\Gamma+c$, $\rho-1$, and Ψ, respectively. The factor $\Gamma+c$ can be written

$$\Gamma+c = -\int_0^{2\pi} \frac{1-\rho}{\rho}\left[\frac{\Gamma}{2\pi} + \frac{\partial\Psi}{\partial r}\right] d\omega - \int_0^{2\pi} \frac{\partial\Psi}{\partial r} d\omega.$$

We again gain a factor $1/c_*{}^2$ from the expression $\rho-1$, and we may obtain

$$\|\Phi(\omega) + \Gamma\omega/2\pi\|_{1+\alpha,\partial D} \leq K_{15}(1/c_*^2 + \|\Psi\|_{2+\alpha}) ,$$

and (5.31) follows. We therefore have an inequality

$$\|N[\Psi,H,c_*]\|_{1+\alpha,\partial D} \leq K_{16}(1/c_*^{\,2} + \|\Psi\|_{2+\alpha})$$

as stated in Lemma A, provided $c_*^{\,2} \geq K_9'/K_1$.

We now consider Lemma B. We wish to verify that if (Ψ_1,H_1), $(\Psi_2,H_2) \in B_{(\delta/2,1)}$ and c_* is large enough, we have

$$(5.32)\quad \|M[\Psi_1,H_1,c_*] - M[\Psi_2,H_2,c_*]\|_\alpha \\ \leq (K_9/c_*^{\,2})(\|\Psi_1-\Psi_3\|_{2+\alpha} + \|H_1-H_3\|_{1+\alpha}) \ ,$$

$$(5.33)\quad \|N[\Psi_1,H_1,c_*] - N[\Psi_2,H_2,c_*]\|_{1+\alpha,\partial D} \\ \leq K_9(\|\Psi_1-\Psi_2\|_{2+\alpha} + \|H_1-H_3\|_{1+\alpha}/c_*^{\,2}) .$$

Consider the first inequality. The form of (5.28) shows that, with an obvious notation, we may write

$$\|M_1 - M_2\|_\alpha \leq \frac{K_{19}}{c_*^{\,2}}\Big(\|\Psi_1-\Psi_2\|_{2+\alpha} + \|H_1-H_2\|_{1+\alpha} + \Big\|\frac{1}{\rho_1^{\gamma+1}} - \frac{1}{\rho_2^{\gamma+1}}\Big\|_\alpha \\ + \|\exp(-2H_1) - \exp(-2H_2)\|_\alpha\Big) \ .$$

It is easily seen that the last term is dominated by some constant times $\|H_1 - H_2\|_{1+\alpha}$. Examination of the expression (5.30) for $\tilde{q}^2$ shows that we also have

$$\|\tilde{q}_1^{\,2} - \tilde{q}_2^{\,2}\|_{1+\alpha} \leq K_{20}\left(\|\Psi_1-\Psi_2\|_{2+\alpha} + \|H_1-H_2\|_{1+\alpha}\right)$$

and using this result with $\rho = R(\tilde{q}^2/c_*^{\,2})$ then gives the first inequality (5.32).

To obtain the estimate (5.33) we write

$$\begin{aligned}\|N_1-N_2\|_{1+\alpha,\partial D} \leq \|&\log\left(1+(2\pi/\Gamma)\partial\Psi_1/\partial r\ (e^{i\omega})\right) \\ &- \log\left(1+(2\pi/\Gamma)\ \partial\Psi_2/\partial r\ (e^{i\omega})\right)\|_{1+\alpha,\partial D} \\ &+\|\log\ \rho_1(e^{i\omega}) - \log\rho_2(e^{i\omega})\|_{1+\alpha,\partial D} \\ &+\|\log\ \hat{Q}(\phi_1(\omega)) - \log\ \hat{Q}(\phi_2(\omega))\|_{1+\alpha,\partial D}\end{aligned}$$

and estimate each of the terms on the right separately. The last expression is the most complicated, and can be treated in the same fashion as was the term $\|\log\ \hat{Q}(\phi(\omega)) - \log\ \dot{Q}(-\Gamma\omega/2\pi)\|_{1+\alpha,\partial D}$ in Lemma A. Straightforward calculations then show that an estimate of the form (5.33) holds, and the statements of Lemma B follow. The map $(\Psi_2,H_2) = L(\Psi_1,H_1,c_*)$ is a contraction and has a unique fixed point that can be obtained by iteration.

There remains a technical point due to the introduction of the factor c in the expression (5.30) for $\phi(\omega)$. We desire the fixed point (Ψ,H) of the mapping to provide

solutions $\psi(\zeta)$ and $F(\zeta) = -\zeta^2 f'(\zeta)$ to the equations (5.11) and (5.13), where (5.14) rather than (5.30) is the expression for $\Phi(\omega)$. Therefore we should check that for the fixed point (Ψ, H) we have

$$c = -\int_0^{2\pi} \frac{1}{\rho} \left[\frac{\Gamma}{2\pi} + \frac{\partial \Psi}{\partial r}\right] d\omega = -\Gamma$$

To see this, note that the corresponding function $\psi(\zeta) = (\Gamma/2\pi) \log|\zeta| + \Psi(\zeta)$ by construction satisfies the equation

$$(\psi_\xi/\rho)_\xi + (\psi_\eta/\rho)_\eta ,$$

with $\psi(\zeta) \sim \Gamma/2\pi \log|\zeta|$ as $\zeta \to 0$, whether or not $c = -\Gamma$. By Green's theorem we therefore have

$$c = \int_0^{2\pi} -\frac{1}{\rho(e^{i\omega})} \frac{\partial \psi}{\partial r} (e^{i\omega})\, d\omega = \lim_{\varepsilon \to 0^+} \int_0^{2\pi} -\frac{1}{\rho(\varepsilon e^{i\omega})} \frac{\partial \psi}{\partial r}(\varepsilon e^{i\omega})\, \varepsilon\, d\omega$$

$$= -\frac{\Gamma}{\rho(0)} .$$

The expression (5.30) for $\tilde{q}^2$ shows that $\rho(0) = R(\tilde{q}^2(0)/c_*^2) = 1$, so $c = -\Gamma$ as desired. This completes the convergence proof for the subsonic inverse problem.

VI. DESCRIPTION OF THE CODE

In this chapter we explain how to modify the input speed distribution in order to obtain airfoils with given specifications. We also describe the other input parameters necessary for the operation of the design mode.

1. Achieving Design Specifications

We refer again to the speed distribution illustrated in Figure $\overset{3}{2}$. The form of the upper surface distribution determines the amount of wave drag experienced by the airfoil and the growth of the boundary layer along the upper surface. We suggest using a distribution with constant supersonic values over the first portion of the profile, followed by decreasing values along the rest of the surface in accordance with the Stratford criterion $C_{sep} \sim 0.003$ near the trailing edge.

The value of c_* should be determined so that the supersonic zone has the proper size, as discussed in Section 4.1.

The lift of the airfoil is related to the area between the upper and lower surface speed distributions. By varying the lower surface distribution, the lift can be distributed

as desired over the airfoil. The free stream Mach number is also affected by changes in the speed distribution. For example, increasing the magnitude of the velocities along the lower surface will generally decrease the lift and increase M_∞.

The thickness-to-chord ratio of the wing section can be adjusted by varying the slope of the speed distribution near the stagnation point $Q = 0$. Increasing the slope will result in a thinner profile with little change in the lift of the airfoil. The free stream Mach number also increases as the thickness-to-chord ratio is decreased. The vertical separation between the upper and lower surfaces is decreased as the slope is increased.

The relative position of the upper and lower surfaces at the trailing edge can be adjusted by changing the velocity distribution near the tail. The vertical separation is increased by raising the prescribed speed at the tail on both the upper and lower surfaces. This will not have a strong effect on the thickness-to-chord ratio. For most purposes the vertical separation should be around 0.015 so that after removing the boundary layer a gap of around 0.007 remains. The horizontal separation can be adjusted by changing the amount of arc length near the tail.

Finally, we mention that a decrease in the prescribed critical speed c_* will generally increase the size of the

supersonic zone, increase the free stream Mach number, decrease the thickness-to-chord ratio, and increase the vertical separation at the trailing edge. It should have little effect on the lift or horizontal separation at the tail.

When designing an airfoil with given specifications it is advisable to proceed in stages by modifying the pressure distribution in the appropriate areas one at a time so that the effects of each change can be isolated. When beginning a new design it is useful to make the initial runs on a coarse mesh of 40×7 or 80×15 points, where results can be obtained in one or two minutes on the CDC 6600. The finer mesh can then be used to make minor adjustments to the pressure distribution and obtain accurate resolution for the final runs.

2. Operation of The Code

We have included the design modification to program H in such a way that when design parameters are not explicitly specified, they assume default values that do not affect the operation of the analysis routine. The description of the operation of the analysis code given in [4] therefore remains in effect for the new version also. We assume the user is familiar with this description of the analysis routine.

For operation of the design mode, the input speed distribution should be provided on TAPE6 in the format given in Table 7.1, with negative values along the lower surface followed by positive values on the upper surface, as in Figure 3. The arc length s must be monotonically increasing.

Table 7.2 contains the other input parameters necessary for operation in the design mode. These parameters have default values as indicated and can also be specified using standard namelist conventions. We provide some sample commands below to illustrate their use.

The parameter NDES specifies the number of overall iterations to be performed in the design mode. Each iteration consists of NS cycles of flow computations, followed by a new mapping to the unit circle as described in Section 3.3. Since the time required to calculate a new

mapping function is small compared to the time required to find the flow past a profile, we use relatively few cycles of flow computation between each mapping. We generally specify NS = 10, NFAST = 0, and NREXLAX = 1, so that 10 relaxation sweeps are used between each mapping. The results we have presented have all been produced with this choice.

The parameter TSTEP is a relaxation factor for the Fourier coefficients determining the mapping function. The rate of convergence of the overall iteration procedure depends on the value of TSTEP. If TSTEP is too large, the coordinates of the profile may oscillate and the scheme will converge slowly or not at all. When a small amount of artificial viscosity is being used in the flow equations, it is sometimes necessary to use smaller values of TSTEP to avoid abrupt changes in the successive profiles that may cause undesirable shock formation. We have TSTEP $\sim$ 0.2 for the design procedure outlined in Section 4.1.

An example of the control cards and data cards used to design an airfoil on the CIMS CDC 6600 is given below. To improve turnaround time we have found it convenient to break up the job into several shorter jobs rather than one longer job.

The relevant control cards have the form

GETPF(TAPE6=SPEED)	Speed contains Q(s) as in Table 7.1
GETPF(LGO=HDES)	HDES is a compiled version of the modified canalysis code H.
LGO.	
SAVE(TAPE3=COMP1)	Tape3 contains data to continue the computation if desired.
SAVE(TAPE4=COORD1)	Tape4 contains the coordinates of the resulting airfoil in FSYM=1.0 format, and the final pressure distribution.

For an initial run on a crude grid, the first card used as input to the program could be

[$P RN = 0., ALP=0., NDES=1 $]

NDES = 1 puts the code in the design mode. The specified angle of attack with respect to the x and y axis must also be given on this card. The Reynolds number is zero for the inviscid flow calculations. The speed distribution is read from TAPE6 and a plot of Q(s) is provided. (If CSTAR < 0, the program will then terminate.)

The next cards are

[$P NS=-1, ITYP=1 $]

The grid is coarsened from 160×30 to 80×15.

[$P NS=-1, ITYP=1 $]

The gird is coarsened from 80×75 15 to 40×7.

```
[ $P NDES=20, NS=10, NFAST=0,
  NRELAX=1, EPS1=0.5, TSTEP=0.2,
  REM=0.5, ITYP=4, XP=1.0,
  KDES=10 ]
```

Twenty design cycles are performed. After each increment of 10 (KDES) cycles, the coordinates of the resulting airfoil, the Mach number diagram, and a Calcomp plot of the flow are produced. XP=1. causes the desired pressure distribution to be plotted on the graph also; if XP=0. it will not appear. The results of this computation are automatically saved on TAPE3 after NDES cycles.

```
[ $P NS=1, ITYP=-1 $ ]
```

The grid is refined from 40×7 to 80×15.

```
[ $P NDES=20, NS=10, ITYP=4 ]
```

Twenty more design runs are performed on the new grid, and the results rewritten on TAPE3.

```
[ $P ITYP=0, XP=0. ]
```

The computation stops. XP=0. causes the airfoil coordinates to be written on TAPE4. This run takes about 90 CP seconds execution time.

If the airfoil produced by this run does not meet the desired design specifications, Q(s) is changed and the run

is repeated. If the airfoil is satisfactory, the run may be continued on a finer mesh as follows.

```
GETPF(TAPE3 = COMP1)
GETPF(LOG = HDES)
LGO.
SAVE(TAPE3 = COMP2)
SAVE(TAPE4 = COORD2)
```

The data cards are then

```
[ $P NDES = 1, RN = 0., ALP = 0. $]
[ $P NS = -1, ITYP = 1 $ ]
```

Coarsen mesh from 160 30 to 80 15.

```
[ $P NS = 0, ITYP = 1 $ ]
```

Read in data stored on TAPE3 (stored with M×N = 80×15) to continue the computation.

```
[ $P NS=1, ITYP= -1 $ ]
[ $P NDES = 10, NS = 10, ESP1 =1., ITYP = 4 $ ]
[ $P ITYP = 0, XP = 0. $ ]
```

This run takes about 130 CP seconds execution time.

The next run might have

```
[ $P NDES = 1, RN = 0., ALP = 0. $ ]
[ $P NS = 0, ITYP = 1 ]
[ $P NDES = 10, NS = 10, EPS1 = 0.75, ITYP = 4 $ ]
[ $P ITYP = 0, XP = 0. $ ] ,
```

and so on.

If it is desired to do the design in a single run, the data cards might read:

```
[ $P NDES = 1, RN = 0., ALP = 0. $ ]
[ $P NS = -1, ITYP = 1 ]
[ $P NS  = -1, ITYP = 1 ]
[ $P NDES = 20, NS = 10, NFAST = 0,
  NRELAX = 1, EPS1 = 0.5, TSTEP = 0.2,
  REM = 0.5, ITYP = 4, KDES = 10, XP = 1. $ ]
[ $P NS = 1, ITYP = -1 $ ]
[ $P NDES = 20, NS = 10, ITYP = 4 $ ]
[ $P NS = 1, ITYP = -1 $ ]
[ $P NDES = 10, NS = 10, EPS1 = 1.0, ITYP = 4 $ ]
[ $P NDES = 10, NS = 10, EPS1 = 0.75, ITYP = 4 $ ]
[ $P NDES = 10, NS = 10, EPS1 = 0.50, ITYP = 4 $ ]
[ $P ITYP = 0, XP = 0. ]
```

This run takes about 520 CP seconds execution time on the CIMS CDC 6600.

For the present version of the code, a separate run is required to perform a boundary layer correction for the airfoil designed with inviscid theory. Assuming the coordinates and pressure distribution from the design run were stored on TAPE4 = COORD1, the control cards for a boundary layer correction would be

```
GETPF(TAPE3 = COORD1)
GETPF(LGO = HDES)
LGO.
SAVE(TAPE3 = COORD2)
```

The required data cards have the form

[$P RN = 20.E06, XP = -1., PCH = 0.07, PLTSZ=8 0 $]

[$P ITYP = 0, XP = 0.]

COORD2 then contains the corrected coordinates in FSYM = 2.0 format. A plot of the profile is also generated by this run.

Finally, we mention that the parameter XOUT can be used to obtain speed distributions for use in the design mode. The command

[$P XOUT = 1.0]

will cause the speed distribution currently in memory to be written on TAPE3 in the format shown in Table 7.1.

BIBLIOGRAPHY

1. Arlinger, B., "An exact method of two-dimensional airfoil design," Saab TN 67, 1970.

2. Bauer, F., Garabedian, P., and Korn, D., A Theory of Supercritical Wing Sections with Computer Programs and Examples, Lecture Notes in Economics and Mathematical Systems, Vol. 66, Springer-Verlag, New York, 1972.

3. Bauer, F., Garabedian, P., Jameson, A., and Korn, D., Supercritical Wing Sections II, a Handbook, Lecture Notes in Economics and Mathematical Systems, Vol. 108, Springer-Verlag, New York, 1975.

4. Bauer, F., Garabedian, P., and Korn, D., Supercritical Wing Sections III, Lecture Notes in Economics and Mathematical Systems, Vol. 150, Springer-Verlag, New York, 1977.

5. Bers, L., Mathematical Aspects of Subsonic and Transonic Gas Dynamics, John Wiley, & Sons, New York, 1958.

6. Bers, L., John, F., and Schechter, M., Partial Differential Equations, John Wiley & Sons., Inc, New York, 1964.

7. Boersteel, J. W., "Review of the application of hodograph theory to transonic aerofoil design and theoretical and experimental analysis of shock-free aerofoils," Symposium Transsonicum II, Springer-Verlag, New York, 1976, pp. 109-133.

8. Boerstoel, J. W., and Huizing, G. H., "Transonic shock-free aerofoil design by an analytic hodograph method," A.I.A.A. Paper No. 74-539, A.I.A.A. 7th Fluid and Plasma Dynamics Conference, Palo Alto, Calif., June 17-19, 1974.

9. Boerstoel, J. W., and Uijlenhoet, R., "Lifting aerofoils with supercritical shockless flow," ICAS paper no. 70-15.

10. Busemann, A., "The nonexistence of transonic flows," Proc. Symposia Appl. Math., Vol. 4, 1953, pp. 29-39.

11. Carlson, L. A., "Inverse transonic flow calculations using experimental pressure distributions," A.I.A.A.J., Vol. 12, No. 4, April 1974, pp. 571-572.

12. Carlson, L. A., "Transonic Airfoil analysis and design using Cartesian coordinates," Journal of Aircraft, Vol. 13, 1976, pp. 349-356.

13. Cebeci, T., Kaups, K., James, R. M., and Mack, D. P., "Boundary-layer and inverse potential methods for axisymmetric bodies," McDonald Douglas Report No. MDC J7895, March 1978.

14. Courant, R., and Friedrichs, K. O., Supersonic Flows and Shock Waves, Interscience-Wiley, New York, 1948.

15. Frankl, F. I., "On the formation of shock waves in subsonic flows with local supersonic velocities," Akad. Nauk SSR, Prikl. Mat. Mech., Vol. 11, 1947, pp. 199-202.

16. Guderley, G., "On the presence of shocks in mixed subsonic-supersonic flow patterns," Advances in Applied Mechanics, Vol. 3, Academic Press, New York, 1953, pp. 145-184.

17. Hedman, S., "Wings in transonic flow, calculation of pressure or surface slope distribution," FFA AU-1043, February, 1975.

18. Hicks, R. M., Vanderplaats, G. N., Murman, E. M., and King, R. R., "Airfoil section drag reduction at transonic speeds by numerical optimization," NASA TM X-73097, February 1976.

19. Ives, D. C., and Liutermoza, J. F., "Analysis of transonic cascade flow using conformal mapping and relaxation techniques," A.I.A.A. Jour., Vol. 15, 1977, pp. 647-652.

20. Jameson, A., "Iterative solution of transonic flows over airfoils and wings, including flows at Mach 1," C.P.A.M., Vol. 27, 1974, pp. 283-309.

21. Jameson, Antony and Caughey, D. A., "Numerical calculations of the transonic flow past a swept wing," ERDA Research and Development Report COO-3077-136, Courant Inst. Math. Sci., New York Univ., June 1977.

22. Langley, M. J., "Numerical methods for two-dimensional and axisymmetric transonic flows," ARA memo 143, 1973.

23. Lighthill, M. J., "A new method of two-dimensional aerodynamic design," Rep. Memor. Aero. Res. Comm., Lond., No. 2112, 1945.

24. Ludford, G. S.S., "The behavior at infinity of the potential function of a two-dimensional subsonic compressible flow," J. Math. Phys., Vol. 30, 1951, pp. 131-159.

25. Milne-Thompson, L. M., Theoretical Aerodynamics, Dover Publications, New York, 1973.

26. Morawetz, C. S., "On the nonexistence of continuous transonic flows past profiles I," Comm. Pure Appl. Math., Vol. 9, 1956, pp. 45-68. "II", Ibid., Vol. 10, 1957, pp. 107-131.

27. Murman, E. M., and Cole, J. D., "Calculation of plane steady transonic flows," A.I.A.A. Jour., Vol. 9, 1971, pp. 114-121.

28. Nash, J. F., and Macdonald, A.G.J., "The calculation of momentum thickness in a turbulent boundary layer at Mach numbers up to unity," C.P. No. 963, British A.R.C., 1967.

29. Nieuwland, G. Y., and Spee, B. M., "Transonic airfoils: recent developments in theory, experiment, and design," Annual Review of Fluid Mechanics, Vol. 5, 1973, Annual Reviews, Inc., Palo Alto, Calif., 1973.

30. Nieuwland, G. Y., "Transonic potential flow around a family of quasi-elliptical aerofoil sections," N.L.R. Report TR.T172, 1967.

31. Pearcey, H. H., "The aerodynamic design of section shapes for swept wings," Advan. Aero. Sci., Vol. 3, 1962, pp. 277-322.

32. Schlichting, H., Boundary-Layer Theory, McGraw-Hill, New York, 1968.

33. Sells, C., "Plane subcritical flow past a lifting airfoil," Proc. Roy. Soc. London, Vol. A308, 1968, pp. 377-401.

34. Steger, J. L., and Klineberg, J. M., "A finite difference method for transonic airfoil design," A.I.A.A. J., Vol. 11, No. 5, May 1973, pp. 628-635.

35. Stratford, B. S., "The prediction of separation of the turbulent boundary layer," J. Fluid Mech., Vol. 5, 1959, pp. 1-16.

36. Tranen, T. L., "A rapid computer aided transonic airfoil design method," A.I.A.A. Paper No. 74-501, June 1974.

TABLE 7.1. TAPE6 INPUT SPEED DISTRIBUTION Q(s).

Cards \ Cols.	1-10	11-20	21-40
1	XIN	CSTAR	
2	$-\lvert Q\rvert$ at tail		initial value of arclength
3	speed along profile		increasing values of arclength
⋮			
XIN + 1	$\lvert Q\rvert$ at tail		final value of arclength

TABLE 7.2. DESIGN PARAMETERS

Glossary of Input Parameters for Design Mode

Parameter	Default	Description
CSTAR	100.	Real. Critical speed c_*. If $c_* < 0$, the program plots the prescribed speed distribution and halts.
KDES	10	Integer. Graphs and flow printout are generated every KDES design cycles.
EPS1	0	Real. Artificial viscosity coefficient ε_1 appearing in the expression (3.16).
NDES	- 1	Integer. Number of design iterations to be performed.
PLTSZ	50.	Real. Length in inches of profile in graph generated when boundary layer correction is performed.
QPL	.85	Real. Lower limit M_0 of the cutoff function V(M) in formula (3.16)
QPU	.95	Real. Upper limit M_1 of the cutoff function V(M) in formula (3.16)
REM	0.	Real. Relaxation parameter for determination of M_∞.
TSTEP	.2	Real. Relaxation parameter for coefficients of mapping function.
XOUT	0.	Real. XOUT = 1 causes the current velocity distribution to be written on TAPE3 in the format of Table 7.1. The computation is then terminated
XIN	None	Real. Number of points used in prescribing input speed distribution Q(s).

Figure 1

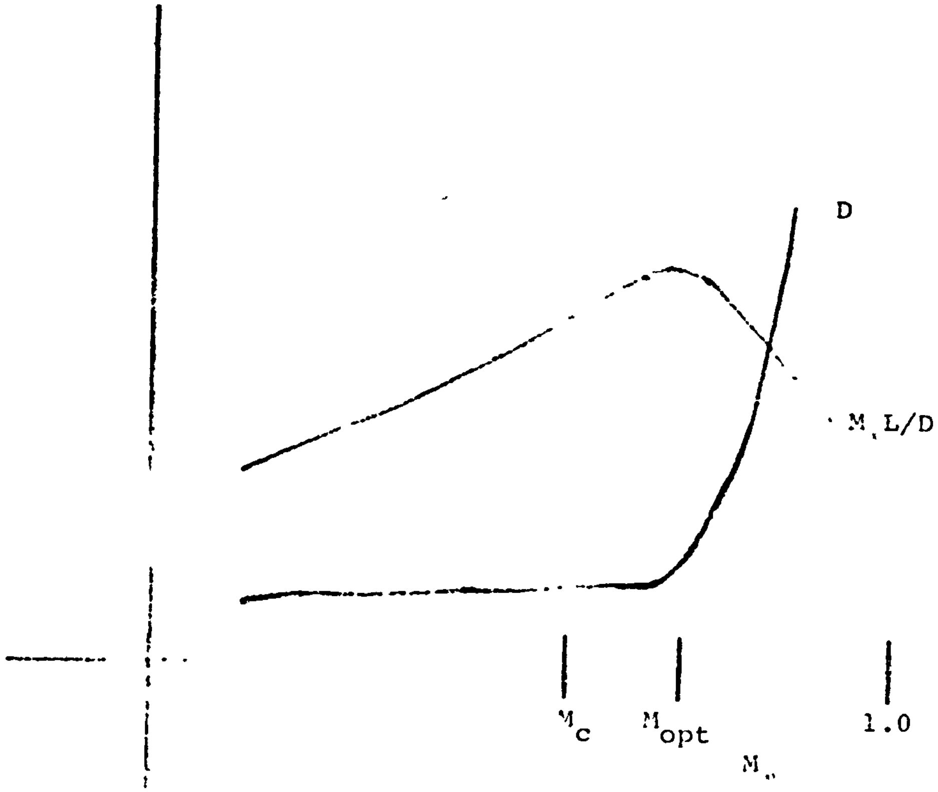
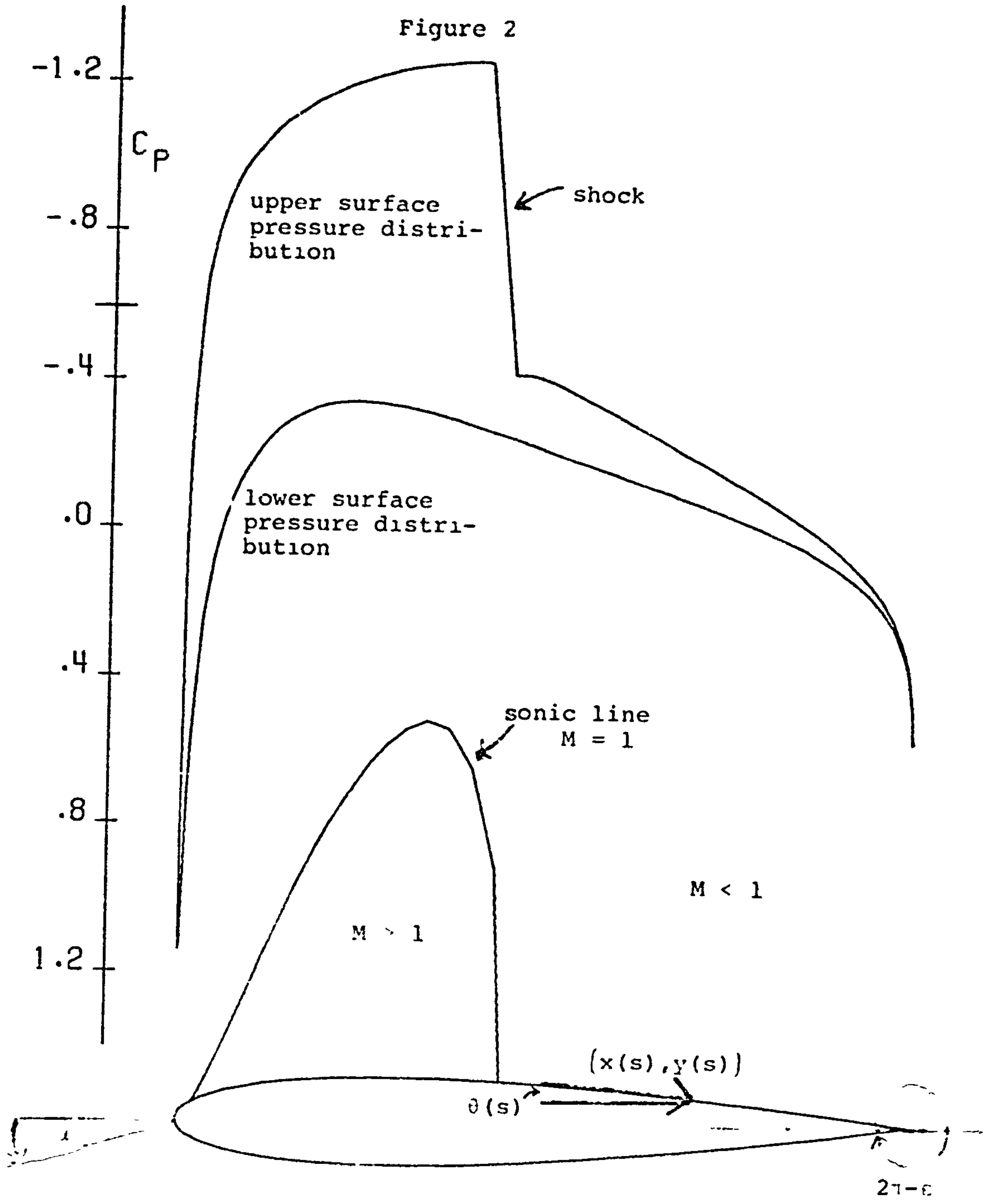


Figure 2



NACA 0012 M*N=160*30 NCY= 10 NO VISCOSITY

— ANALYSIS M=.750 ALP= 2.00 CL= .445 CD=.0057

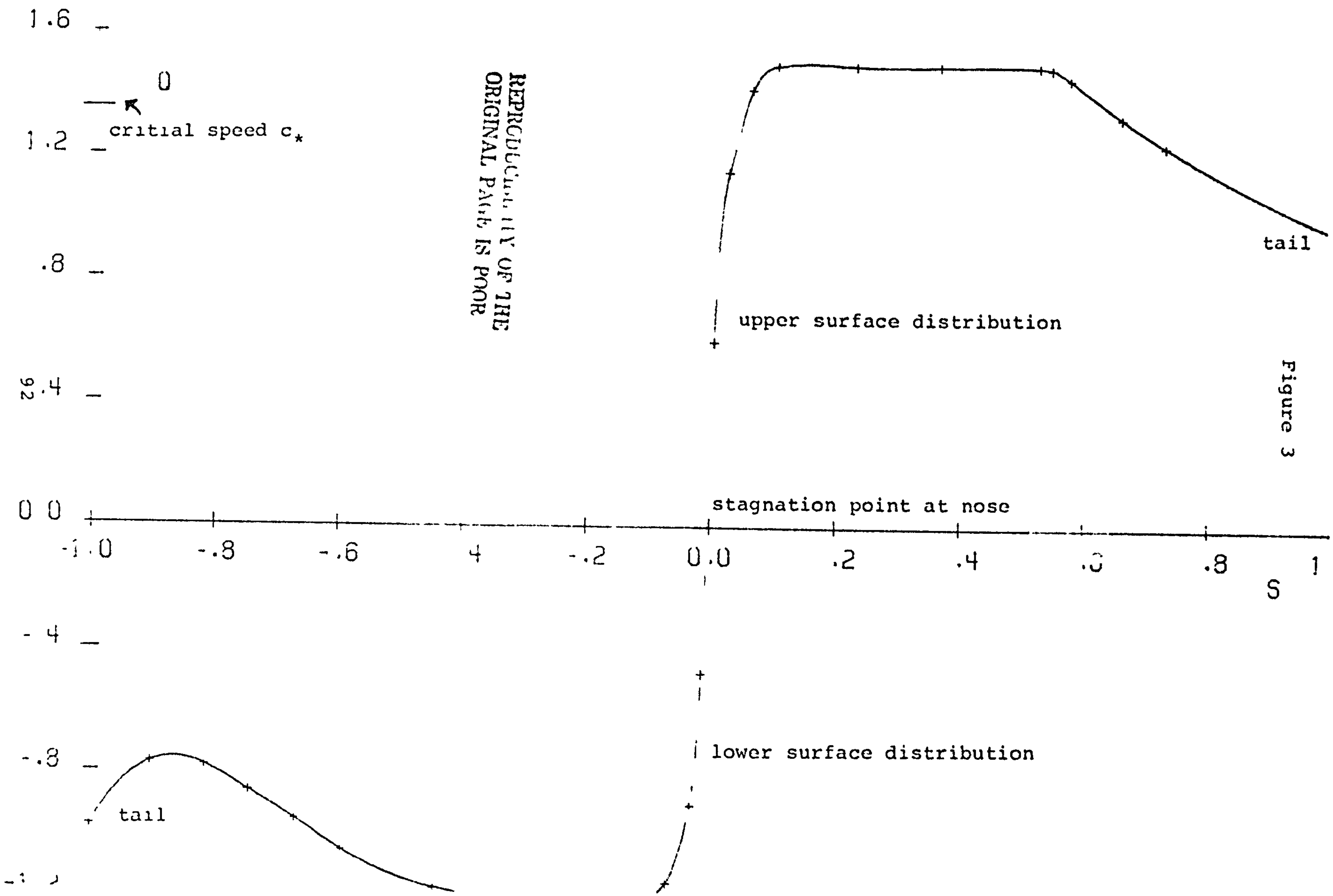
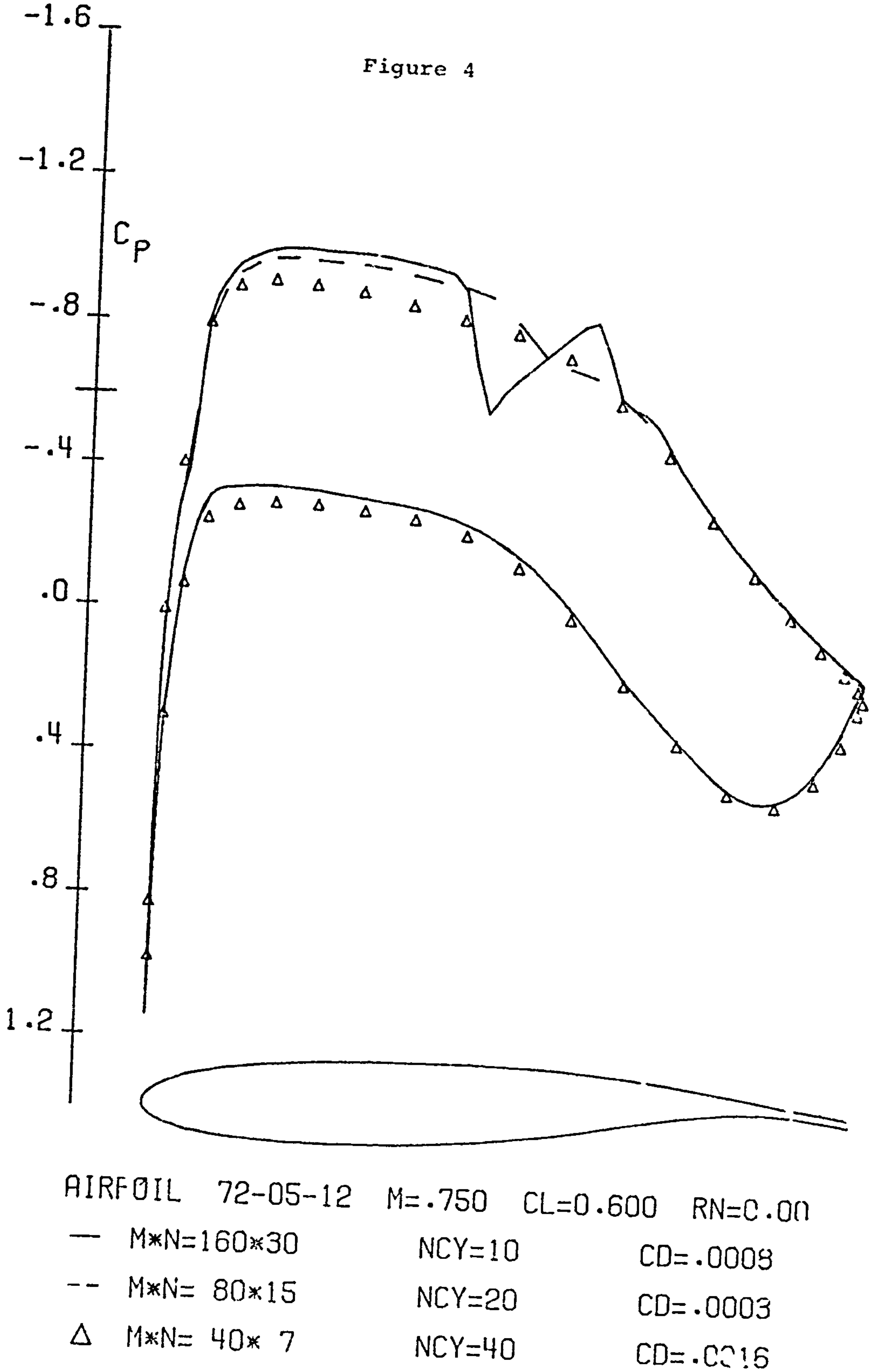

1.6
1.2
.8
.4
0.0
-.4
-.8
critical speed c_*
upper surface distribution
lower surface distribution
stagnation point at nose
tail
tail
-1.0
-.8
-.6
-.2
0.0
.2
.4
.8
1
S

Figure 3

Figure 4



-1.6
-1.2
CP
-.8
-.4
.0
.4
.8
1.2
AIRFOIL 72-05-12 M=.750 CL=0.600 RN=0.00
— M*N=160*30 NCY=10 CD=.0008
-- M*N= 80*15 NCY=20 CD=.0003
Δ M*N= 40* 7 NCY=40 CD=.0016

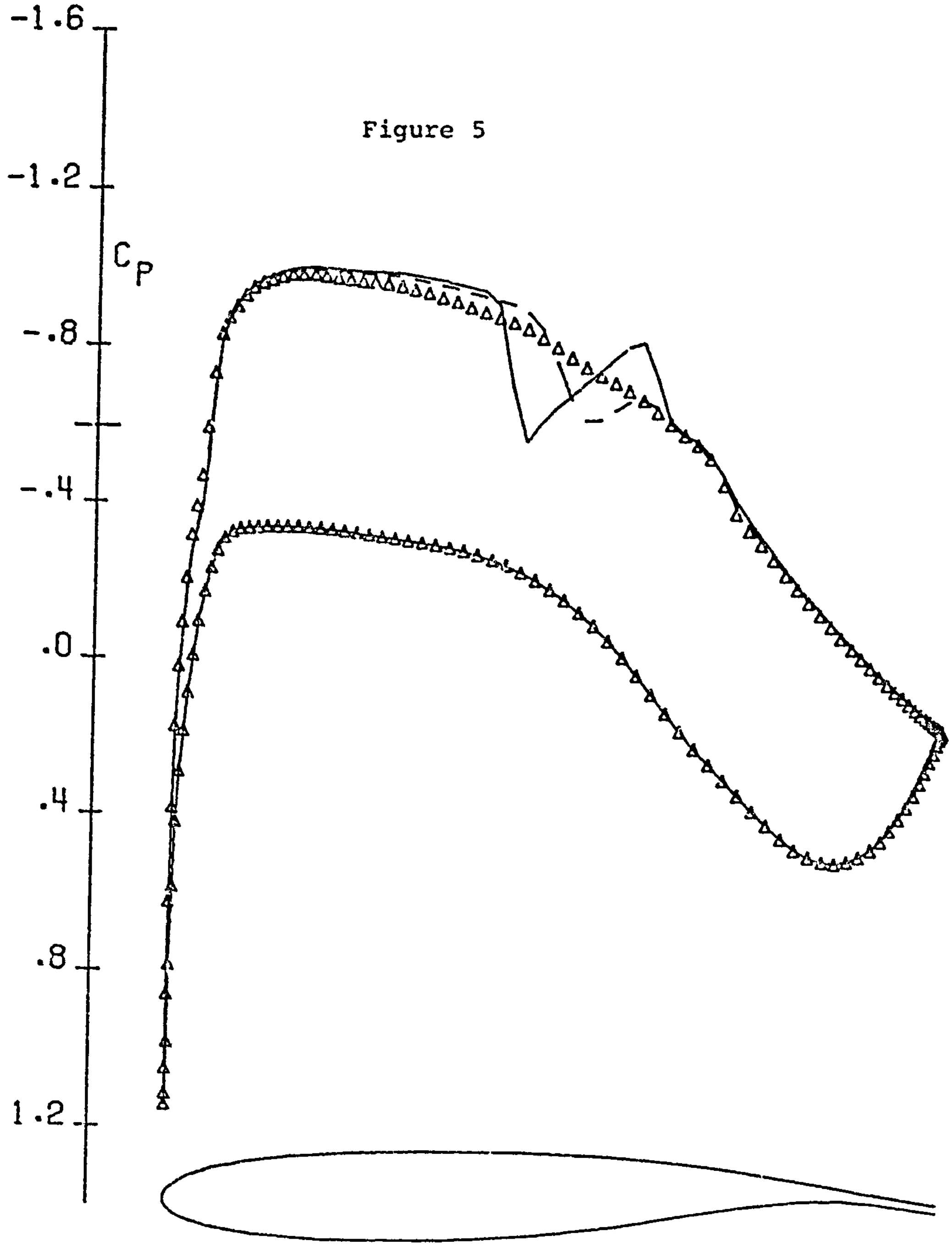


Figure 5

AIRFOIL 72-05-12 M=.750 CL=0.600 RN=0.00

—	M*N=160*30	EPS1=0.0	CD=.0008
--	M*N=160*30	EPS1=0.1	CD=.0002
△	M*N=160*30	EPS1=0.4	CD=.0000

C-2

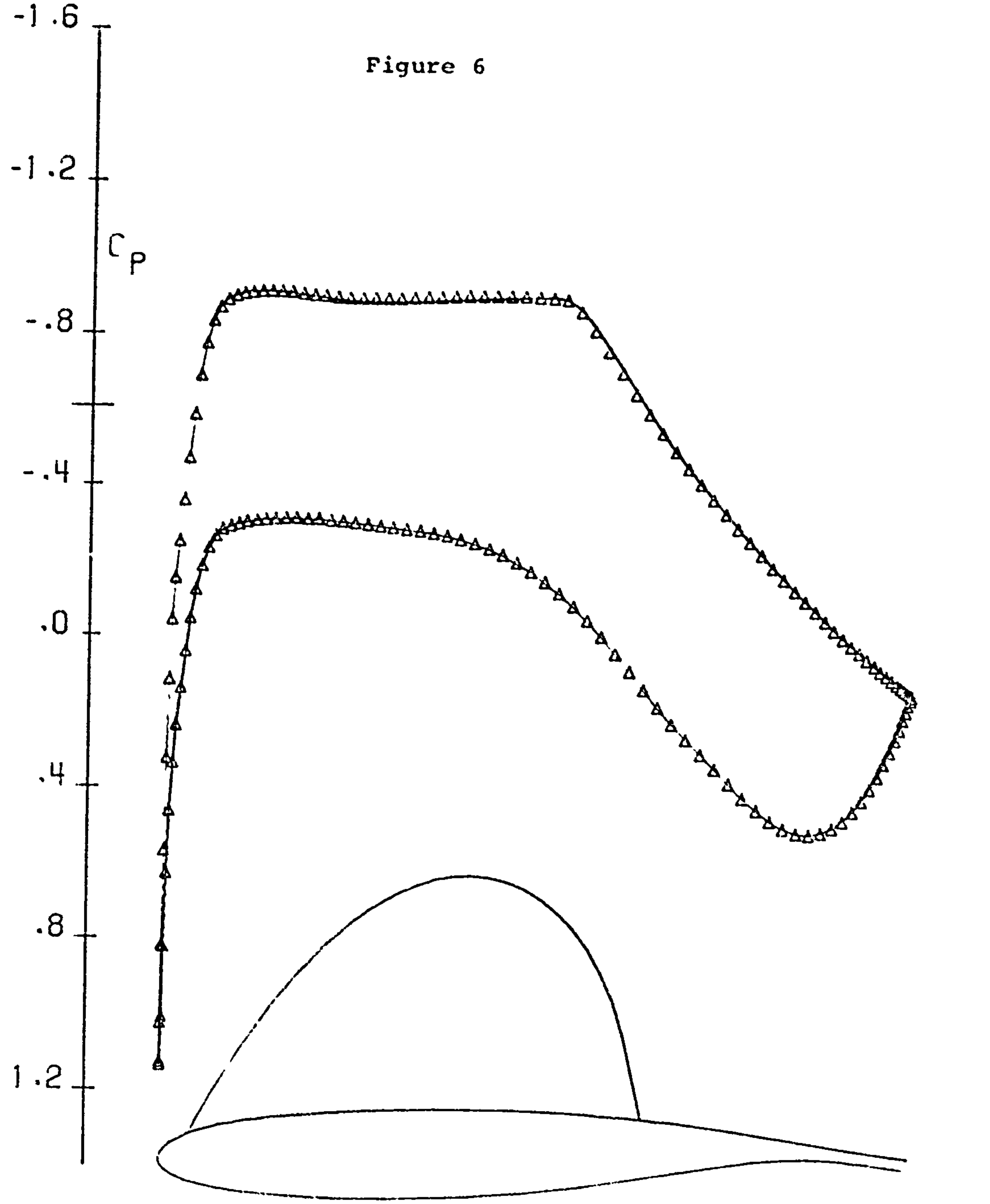


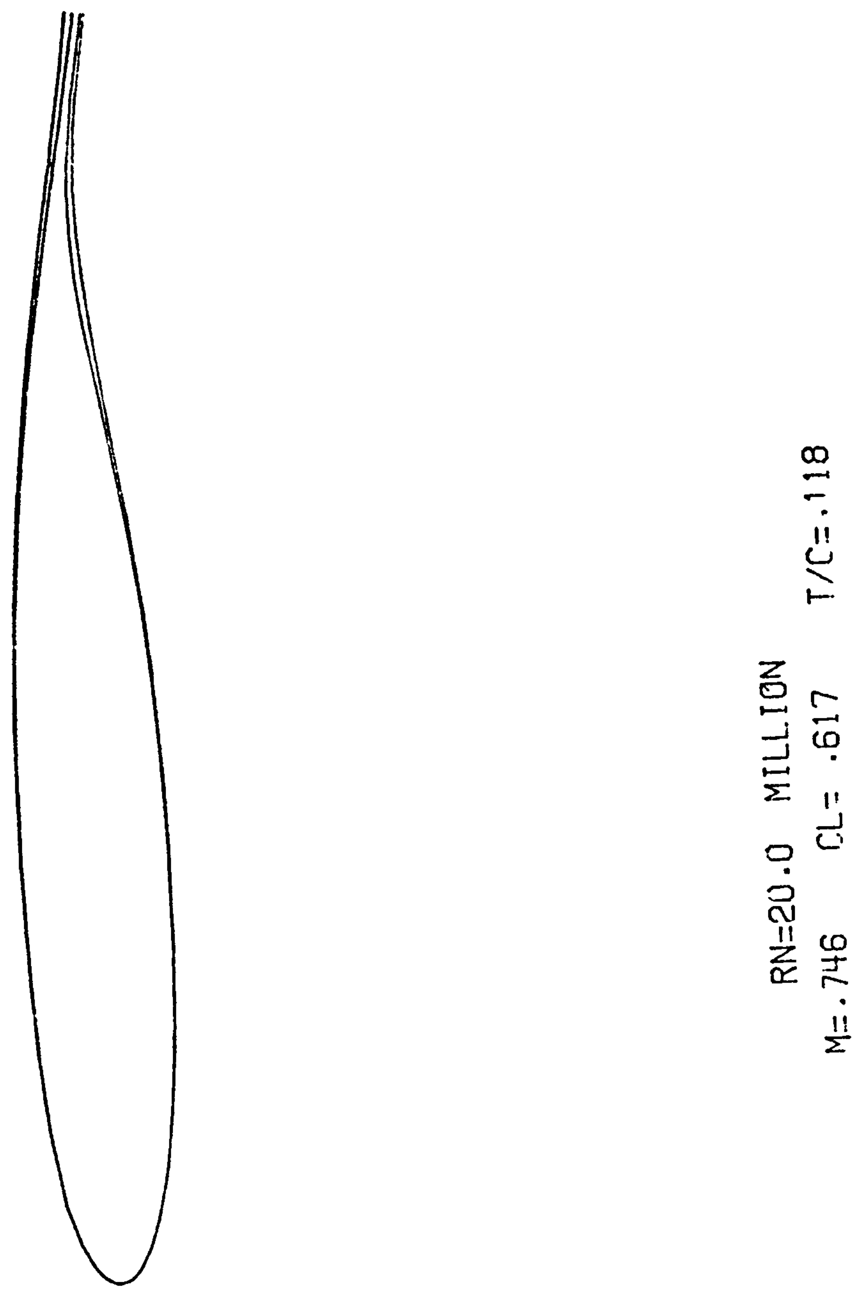
Figure 6

VISCOUS DESIGN M*N=160*30 NCY= 300 EPS1= .500

— ART. VISC. M=.746 ALP= 0.00 CL= .613 CD=.0011

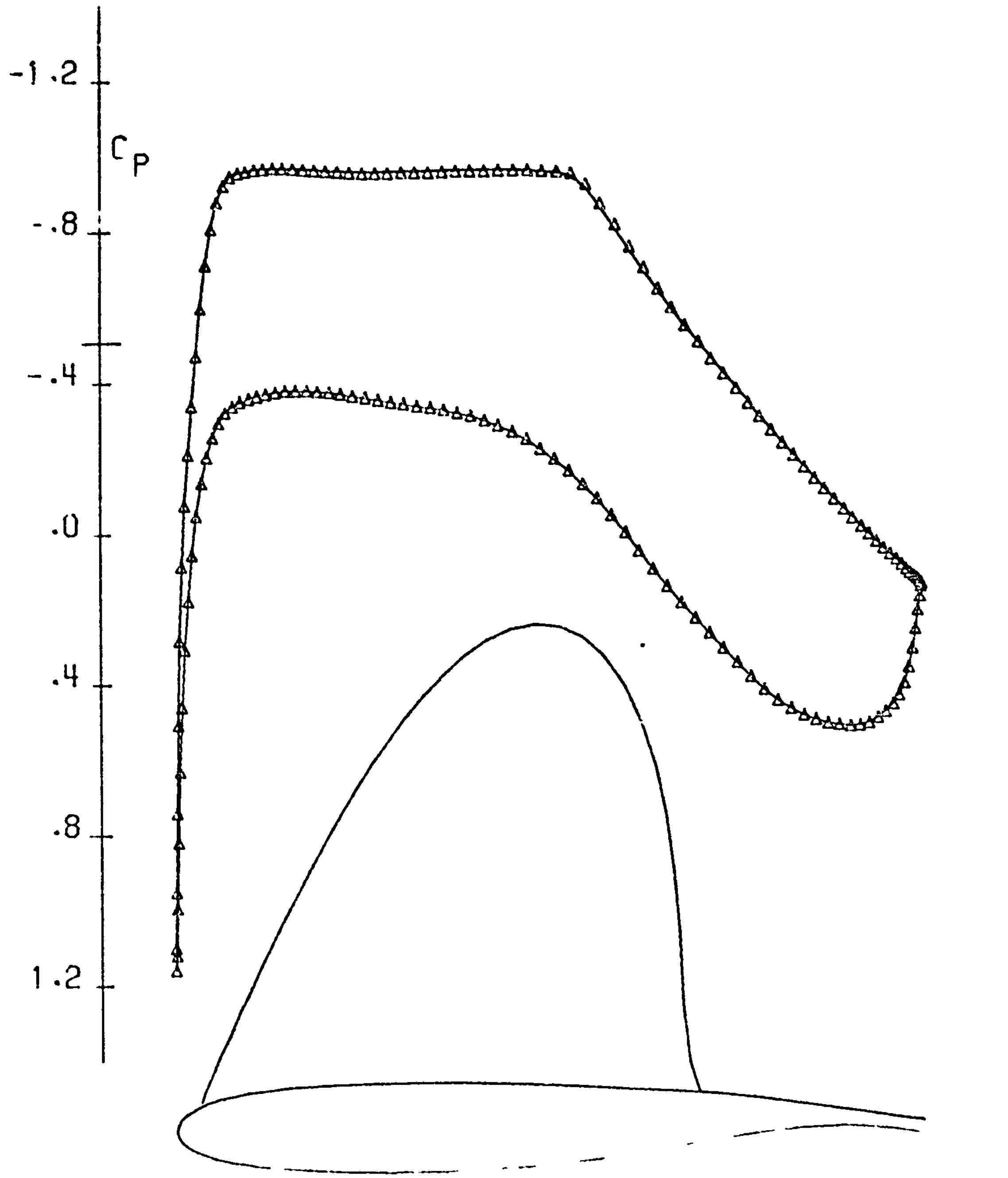
Δ INPUT CP T/C= .118 DO= .64E-02 DPHI= .11E-03

Figure 7



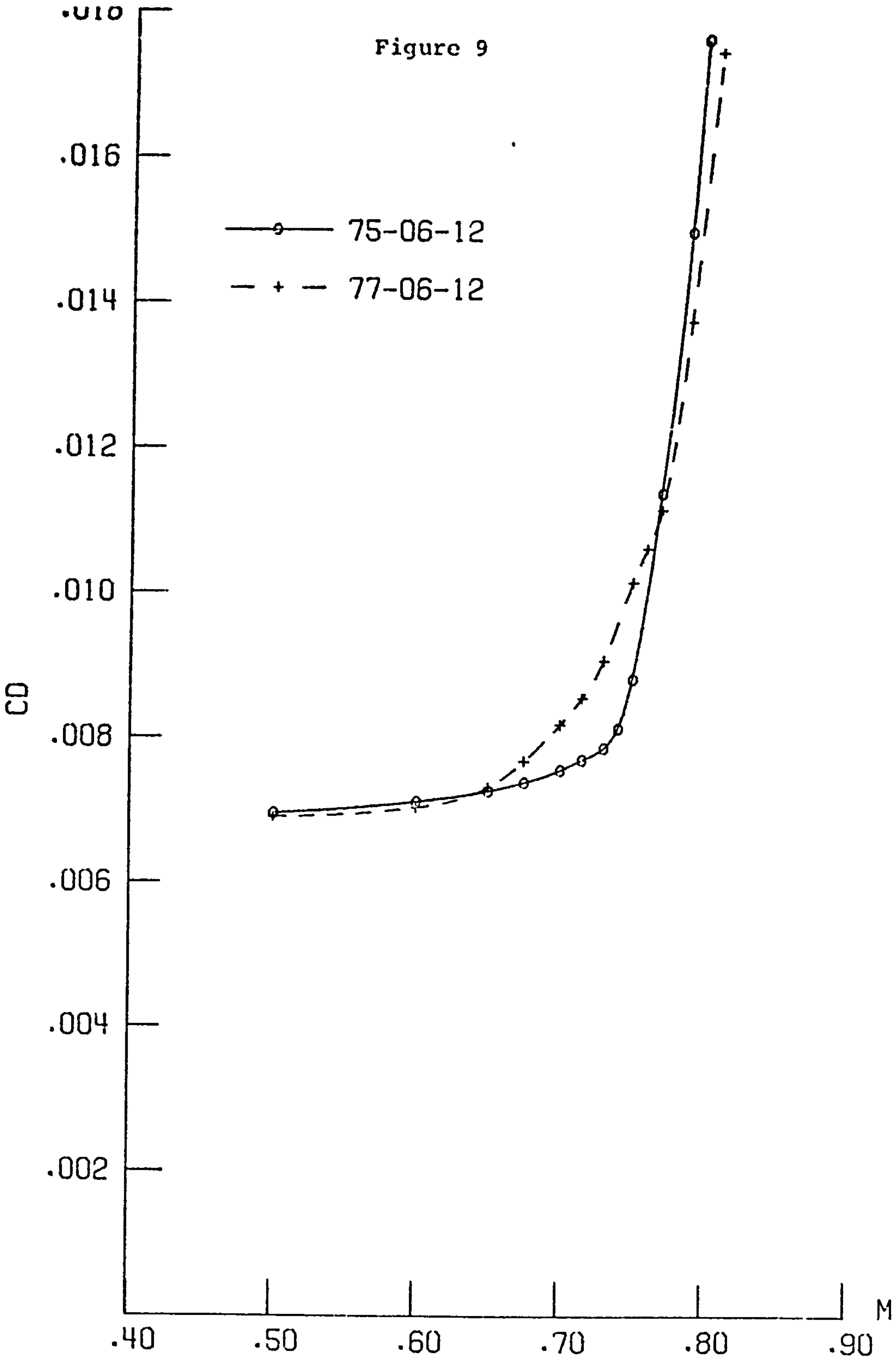
REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

Figure 8



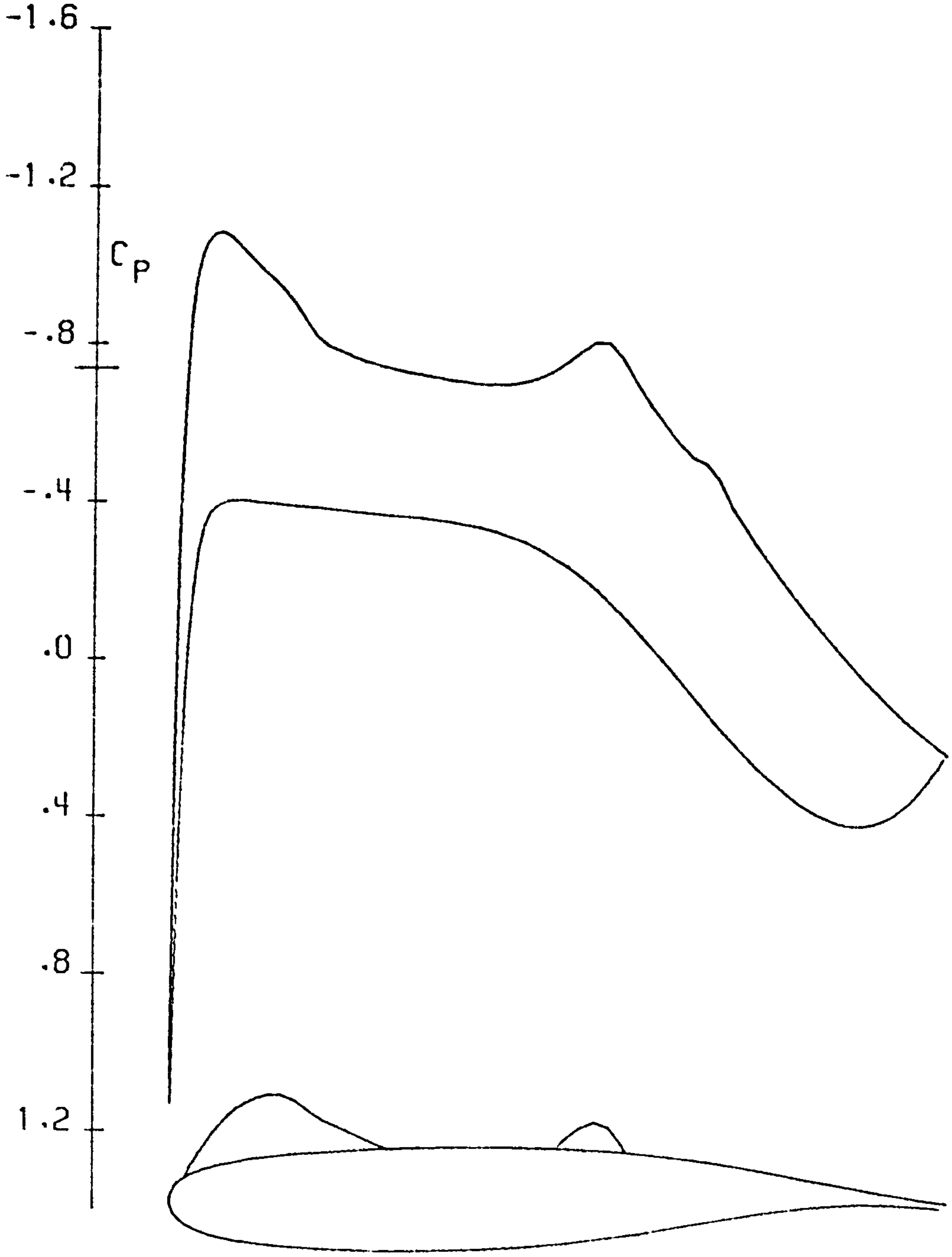
VISCOUS DESIGN M*N=160*30 NCY= 200 EPS1= .750
— ART. VISC. M=.776 ALP= 0.00 CL= .639 CD=.0056
Δ INPUT CP T/C= .118 DQ= .45E-02 DPHI=-.49E-04

Figure 9



DRAG RISE CURVES FOR AIRFOIL 75-06-12 AND AIRFOIL 77-06-12, CL=0.61, RN=20 MILLION

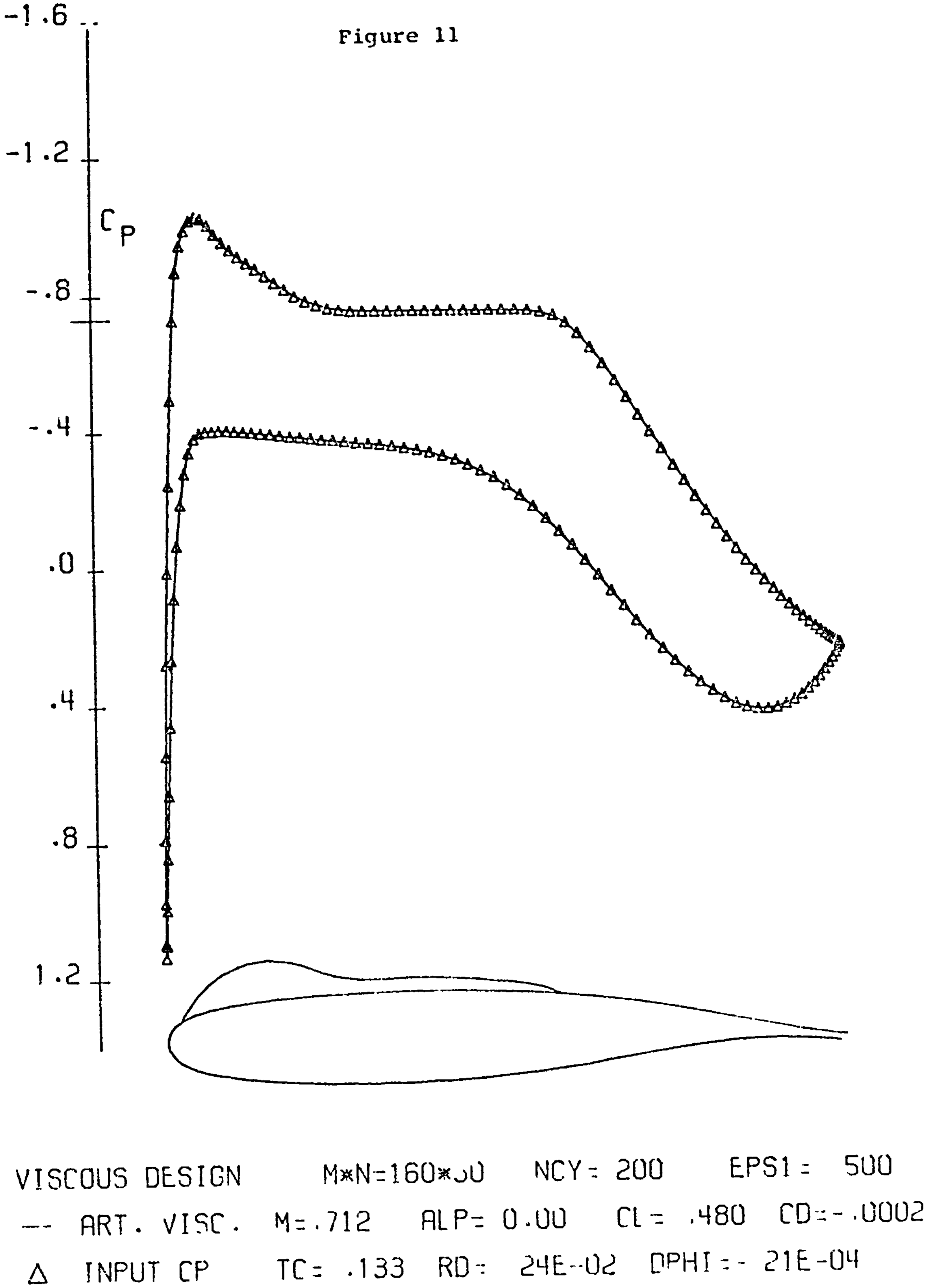
Figure 10



AIRFOIL 74-06-13 M*N=160*30 NCY= 10 NO VISCOSITY

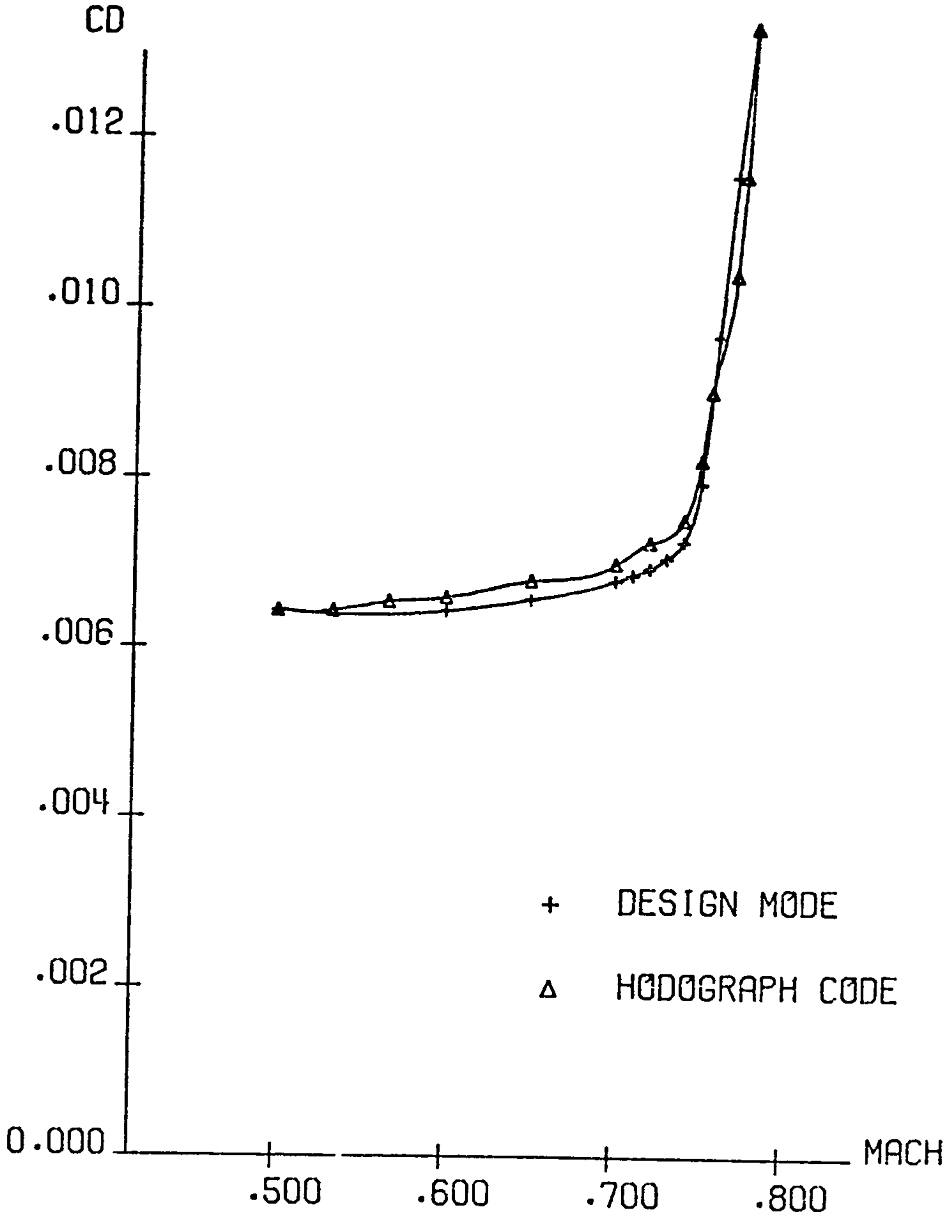
— ANALYSIS M= 710 ALP= 0.00 CL= .481 CD=-.0001

Figure 11



-1.6
-1.2
-.8
-.4
.0
.4
.8
1.2
C_P
VISCOUS DESIGN M*N=160*30 NCY= 200 EPS1= 500
-- ART. VISC. M=.712 ALP= 0.00 CL= .480 CD=-.0002
Δ INPUT CP TC= .133 RD= 24E-02 DPHI=- 21E-04

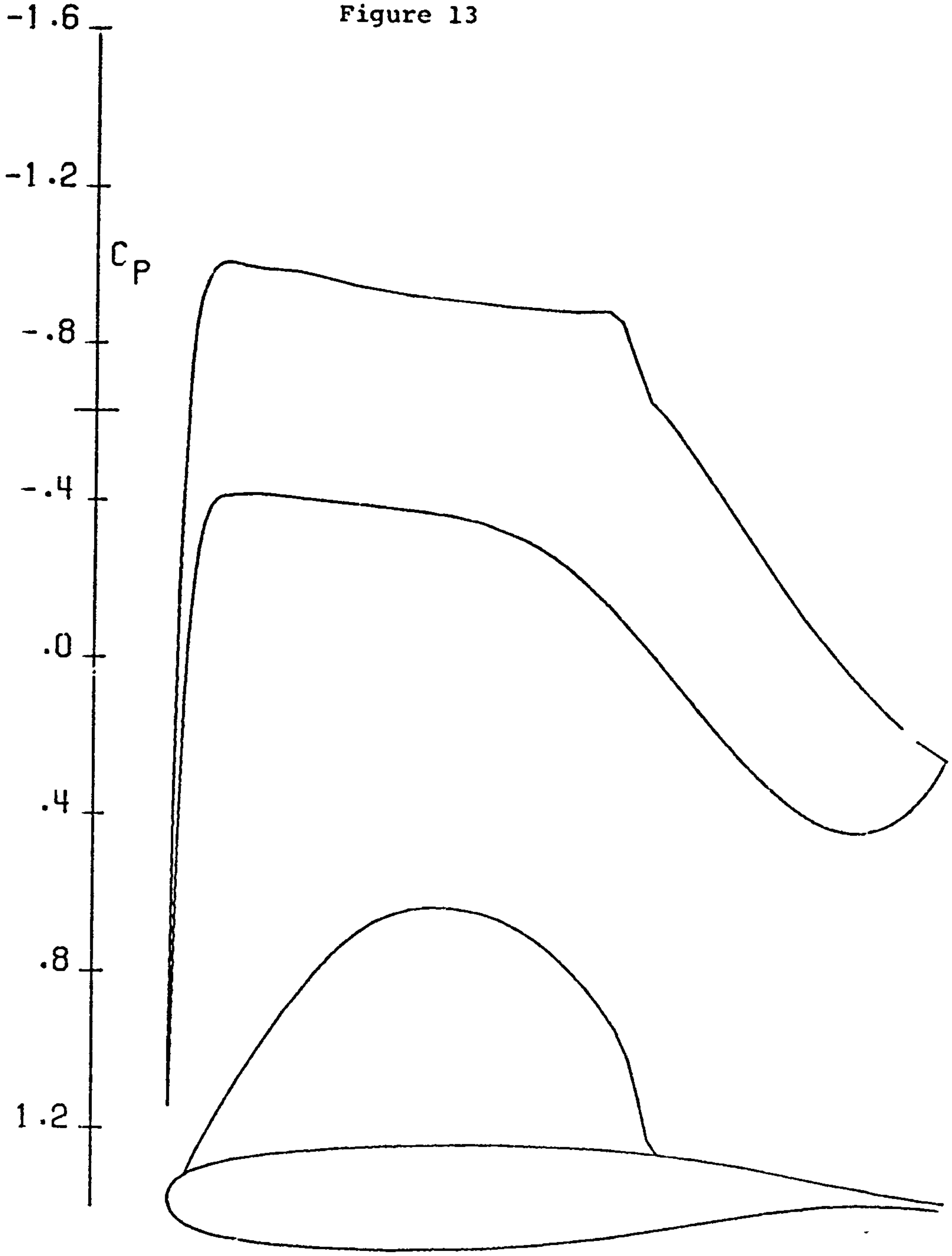
Figure 12



DRAG RISE CURVES FOR SUPERCRITICAL

AIRFOILS AT CL=.54 AND R = 8 MILLION

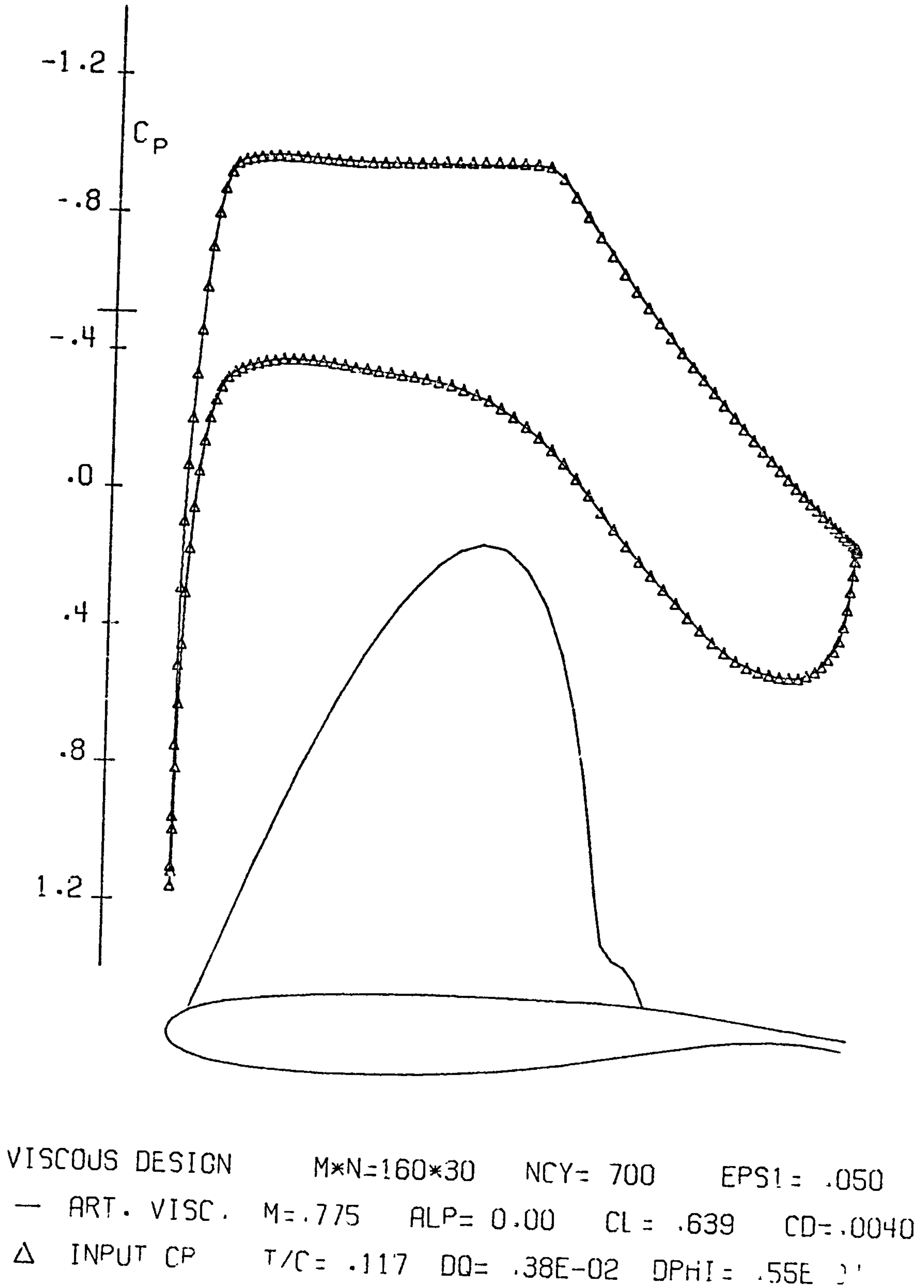
Figure 13



AIRFOIL 75-05-13 M*N=160*30 NCY= 10 R= 8 MILLI

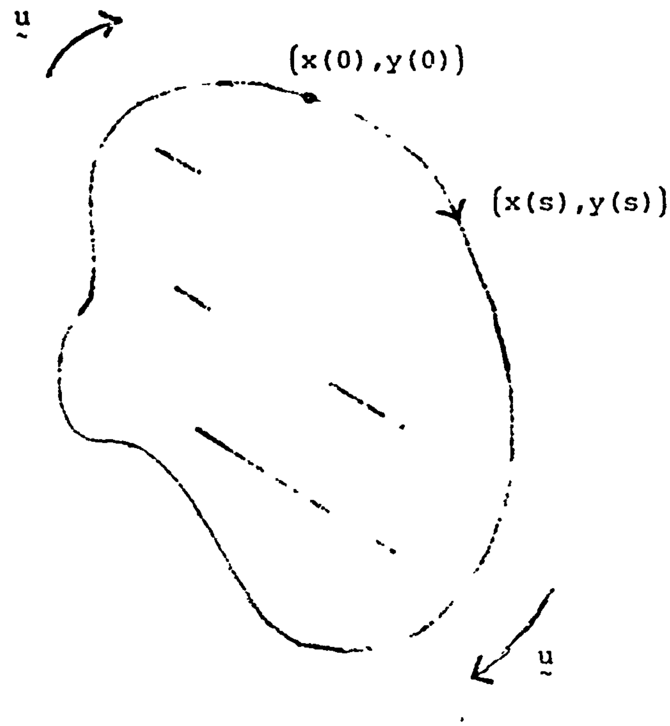
— THEORY M=.740 ALP= .01 CL= .540 CD=.00

Figure 14



-1.2
C_P
-.8
-.4
.0
.4
.8
1.2
VISCOUS DESIGN M*N=160*30 NCY= 700 EPS1= .050
— ART. VISC. M=.775 ALP= 0.00 CL= .639 CD=.0040
Δ INPUT CP T/C= .117 DQ= .38E-02 DPHI= .55E

Figure 15



LISTING OF CODE

```
      PROGRAM H(INPUT = 66,OUTPUT = 500,TAPE3 = 600,TAPE4 = 400,TAPE2 =
     1OUTPUT,TAPE5 = INPUT,TAPE6)
      COMMON/FL/FLUXT4,CD4,COW,INDCD
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),FPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,FN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TL,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NPN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      DIMENSION COMC(87),CLA(2),NAMERR(6)
      EQUIVALENCE (COMC(1),PI),(CLX,CLA(1)),(ALPX,CLA(2))
C     LSTERR IS THE SUBROUTINE TO PROCESS A NAMELIST ERROR
      EXTERNAL LSTERR
C     ***NON-ANSI***
      NAMELIST /P/ ALP,BETA,BCP,CL,EM,FSYM,GAMMA,IS,ITYP,IZ,KP,LL,LSEP,
     1   M,N,NFC,NPTS,NRN,NS,NS1,PCH,RBCP,RCL,RDEL,RFLO,RN,SEPM,ST,
     2 XMON,XP,XSEP,NRELAX,NFAST
     3 ,EPS1,CSTAR,NDES,REM,DEP,TSTEP,XOUT,KDES,PLTSZ,QPL,QPU
      DATA GAMMA/1.4/ , ST/0./ , XMON/.95/  , RBCP/.10/ , RFLO/1.4/ ,
     1              RDEL/.125/ , BCP/.4/ , NS1/20/ , NS/1/ , KP/1/
      DATA N5/5/ , NAMERR/6*0/ , D1,D2,SL/3*0./ , CP1/.4/ ,XPF/1./
     1  ,N6/6/
      AA(1) = 99999.
      INDCD=0
      NEW=1
      NFAST=1
      NRELAX=6
C     THESE TWO CARDS TRANSMIT TO THE SYSTEM THE RECOVERY ADDRESS
      NAMERR(5) = LOCF(LSTERR)
      CALL SYSTEMC(66,NAMERR)
      M4 = N4
      REWIND N4
      WRITE (N2,180)
      READ (N5,P)
      IF (CL.NE.100.) MODE = 0
```

```
      IF (IZ.GE.80) N4 = N2
      IF (NS.EQ.0) GO TO 30
      CALL RESTRT
      CLX = CL
      ALPX= RAD*ALP
      GO TO 140
   10 WRITE (N2,180)
      NEW=1
      ALP = 100.
      CL = 100.
C     ****NON-ANSI****
      READ (N5,P)
      LN = RN*1.E-6+.5
      TXT = 3HALP
      IF (MODE.EQ.0) TXT - 3H CL
      CALL SECOND(TIME)
      WRITE(N2,200) EM,TXT,CLA(MODE+1),LN,M,N,NS,TIME,RFLO,RCL,RDEL,
     1 RBCP,BETA,ST,PCH,SEPM,XSEP,NPTS,IS,LL,IZ
     2 ,EPS1,NDES,REM,NFAST,NRELAX,ITYP,FSYM,TSTEP,DEP
      IF (ABS(XOUT).GT..5) GO TO 7727
      GO TO 7728
 7727 CALL OUTPT
      CALL PLOT(0.,0.,999)
      STOP
 7728 CONTINUE
C     SELECT OUTPUT TAPE
      N4 = M4
      IF (IZ.GE.80) N4 = N2
      C2 = .5*(GAMMA-1.)
      C7 = GAMMA/(GAMMA-1.)
      IF (ALP.EQ.100.) GO TO 20
C     ALP HAS BEEN INPUTTED, KEEP IT FIXED
      NCY = 0
      MODE = 1
      ALPX = ALP
   20 ALP = ALPX/RAD
      IF (CL.EQ.100.) GO TO 25
C     CL HAS BEEN INPUTTED, KEEP IT FIXED
      NCY = 0
      MODE = 0
      YA = .5*CL/CHD-DPHI
      DO 114 L = 1,M
      DO 114 J = 1,NN
  114 PHI(L,J) = PHI(L,J)+YA*PHIR(L)
      DPHI = .5*CL/CHD
      CLX = CL
   25 CL = CLX
C     CHANGE PARAMETERS WHICH DEPEND ON THE MACH NUMBER
      EM  = AMAX1(EM,.1E-40)
      IF (EM.NE.EMX) NCY = 0
      C1 = C2+1./(EM *EM )
      C6 = C2*EM *EM
      C4 = 1.+C6
      C5 = 1./(C6*C7)
```

```
      QCRIT = (C1+C1)/(GAMMA+1.)
      BET = SQRT(1.-EM *EM )-1.
C     CHECK FOR TERMINATE,RETRIEVE, OR STORE INSTRUCTIONS
C     IK WILL BE -1 ONLY IF THERE IS A NAMELIST ERROR
      IF ((ITYP.EQ.0).OR.(IK.EQ.-1)) GO TO 170
      CALL COSI
      IF (NS.NE.0) GO TO 40
      REWIND N3
      IF (ITYP.GT.0) GO TO 30
      WRITE(N3) COMC,PHI,AA,BB,ARCOLD,ANGOLD,XOLD,YOLD,DELOLD,K,RS,RI
     1 ,DSUM,GAMMA,XMON,RBCP,RFLO,RDEL,BCP,NS1,KP,ST
      GO TO 140
   30 READ (N3) COMC,PHI,AA,BB,ARCOLD,ANGOLD,XOLD,YOLD,DELOLD,K,RS,RI
     1 ,DSUM,GAMMA,XMON,RBCP,RFLO,RDEL,BCP,NS1,KP,ST
      CALL MAP
      GO TO 140
   40 CONTINUE
      IF (NS.GT.0) GO TO 70
      NS = 0
C     GO TO CRUDE GRID IF ITYP.GT.0
      IF (ITYP.GT.0) CALL REMESH(-1)
      GO TO 140
   70 IF (ITYP.GT.0) GO TO 99
C     GO BACK TO FINER GRID
      CALL REMESH(1)
      GO TO 140
C     SET UP CONSTANTS AND DO CONFORMAL MAPPING
   99 KD = 1
  100 XPHII = 0.
      IF ( RCL.NE.0.) XPHII = 2.*CHD/RCL
      XA = 1.-2./RFLO
      ANGO = -RAD*BB(1)
      TXT = 3H CL
      IF (MODE.EQ.0) TXT = 3HALP
C     DO AT MOST NS CYCLES
      IF (RN.LE.0.) NS1 = 1000000
      IXX = M+2
   80 IXX = IXX-1
      IF (XC(IXX-1).GT.XMON) GO TO 80
      LC = 0
      DO 120 K = 1,NS
      IF (NDES.GE.0) GO TO 105
      IF (MOD(LC,56).NE.0) GO TO 105
      WRITE (N2,210) TXT
      LC = LC+1
  105 CONTINUE
      IF(NFAST.LE.0) GO TO 141
      CALL SWEEP1
  141 IF(NRELAX.LE.0) GO TO 151
      DO 142 LF=1,NRELAX
      CALL SWEEP
  142 CONTINUE
  151 NEW=0
      NCY = NCY+1
```

```
      ALPX = RAD*ALP
      CLX= 2.*DPHI*CHD
      YA = YA*XPHII
C     WRITE RESIDUALS ON N2 EVERY KP CYCLES
      IF (NDES.GE.0) GO TO 110
      IF (MOD(K,KP).NE.0) GO TO 110
      LC = LC+1
      INDCD=1
      CALL GTURB(D1,D2,CP1,CDW,SL,RDEL,RBCP)
      INDCD=0
      ANGO = -RAD*BB(1)
      WRITE (N2,190) NCY,YR,YA,D1,D2,IK,JK,NSP,CLA(2-MODE),ANGO,CP1,
     1 CDW,CD4
C     DO A BOUNDRY LAYER CORRECTION EVERY NS1 CYCLES
  110 IF (MOD(K,NS1).NE.0) GO TO 125
      IF (K.EQ.NS) GO TO 140
      WRITE (N2,190)
      LC = LC+1
      FSYM = 6.
      CALL GTURB(D1,D2,CP1,BCP,SL,RDEL,RBCP)
      ANGO = -RAD*BB(1)
      IF (MODE.EQ.0) DPHI = .5*CLX/CHD
C     CHECK TO SEE IF WE HAVE SATISFIED CONVERGENCE CRITERIA
  125 IF (AMAX1(ABS(YR),ABS(YA)).LT.ST) GO TO 310
  120 CONTINUE
  310 IF(NDES.LE.0) GO TO 140
      CALL CYCLE
      IF (MOD( KD,KDES ).NE.0) GO TO 138
      CALL GTURB(D1,D2,CP1,BCP,SL,RDEL,RBCP)
      CALL MAP
      CALL COSI
  138 IF (KD.EQ.NDES) GO TO 139
      KD = KD+1
      GO TO 100
  139 REWIND N3
      ITYP = 1
      WRITE(N3) COMC,PHI,AA,BB,ARCOLD,ANGOLD,XOLD,YOLD,DELOLD,R,RS,RI
     1 ,DSUM,GAMMA,XMON,RBCP,RFLO,RDEL,BCP,NS1,KP,ST
  140 ITYP = IABS(ITYP)
      CL = CLX
      LN = RN*1.E-6+.5
      XPF = XPF*AMINO(1,IABS(M4-N4))
      XP = XP*XPF
      CALL SECOND(TIME)
      NTPE = N4
      TXT = 3HALP
      IF (MODE.EQ.0) TXT = 3H CL
  150 WRITE (NTPE,200) EM,TXT,CLA(MODE+1),LN,M,N,NS,TIME,RFLO,RCL,RDEL,
     1 RBCP,PETA,ST,PCH,SEPM,XSEP,NPTS,IS,LL,IZ
     2 ,EPS1,NDES,REM,NFAST,NRELAX,ITYP,FSYM,TSTEP,DLP
      IF (NTPE.EQ.N2) GO TO 160
      NTPE = N2
      GO TO 150
  160 IF (ITYP.GE.2) CALL GTURB(D1,D2,CP1,BCP,SL,RDEL,RBCP)
```

```
      EMX = EM
      ITYP=1
      GO TO 10
  170 ITYP = 4
      IF (IK.EQ.-1) WRITE (N4,220)
      CALL GTURB(D1,D2,CP1,BCP,SL,RDEL,RBCP)
C     TERMINATE PLOT
      CALL PLOT(0.,0.,999)
      CALL EXIT
  180 FORMAT (7H READ P/)
  190 FORMAT(5X,I4,4F12.3,I4,I3,I6,2F10.4,2F11.5,F11.5)
  200 FORMAT (4H0EM=F4.3,3XA3,1H=F5.2,3X3HRN=I2,2HE6,3X4HM*N=,I3,1H*,I2,
     1 3X3HNS=I4,3X5HTIME=F7.2/6H RFLO=F4.2,3X,4HRCL=F4.2,3X5HRDEL=F4.3
     2 ,3X5HRBCP=F3.2,3X5HLETA=F4.2,3X3HST=,E7.1/ 5H PCH=F4.2,
     3  3X5HSEPM=F5.4,3X5HXSEP=F4.2,3X5HNPTS=I3,3X3HIS=I2,3X3HLL=I3,
     4 3X3HIZ=I3/ 6H EPS1=F8.4,3X6H NDES=I3,3X
     5 ,4HREM=F8.4,7H NFAST=I3,3X,7HNRELAX=I3,/,
     6 3X,5HITYP=I3,3X,5HFSYM=F8.4,3X,6HTSTEP=,F6.4,3X,4HDEP=,F8.4,//)
  210 FORMAT(1H15X3HNCY6X4HDPHI8X3HDCL,8X,4HDDEL,6X,4HDBCP,5X,2HIK,
     1 2X,2HJK,2X3HNSP,5XA4,5X4HANGO,8X3HCPI,8X3HCDW,8X2HCD/)
  220 FORMAT (21H0***NAMELIST ERROR***,10X,20HPROGRAM TO TERMINATE   )
      END



      SUBROUTINE LSTERR
      COMMON /A/ M(47),IK
      IK = -1
      RETURN
      END



      SUBROUTINE RESTRT
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CC(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,DA,DCN,DSN,RA4,EPSIL,CCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TILE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODL,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALO1
     5 ,SCALO3,N6,GAMMA,NOPT,CSTAR,REM,DEP,QINF,TSTEP,XCUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
C     SET UP CONSTANTS
      TP = PI+PI
      RAD = 180./PI
      ALP = ALP/RAD
      IF ((N+1).NE.NN.OR.(M+1).NE.MM) NCY = 0
      MM = M+1
```

```
      IF (LL.EQ.0) LL = M/2+1
      NN = N+1
      DR = -1./FLOAT(N)
      DT = TP/FLOAT(M)
      DCN = COS(DT)
      DSN = SIN(DT)
      DELR = .5/DR
      DELTH = .5/DT
      RA = DT/DR
      RA4 = DT*DT
      DO 10 K = 1,N
      R(K) = 1.+DR*FLOAT(K-1)
      RS(K) = (RA*R(K))*(RA*R(K))
      RI(K) = -.25*DT/R(K)
   10 CONTINUE
      R(NN) = 0.
      BET = SQRT(1.-EM*EM) -1.
      IF (NDES.GE.0) GO TO  5
      CALL AIRFOL
      GO TO 6
    5 CALL READOS
    6 CONTINUE
      IF (MODE.EQ.1) CL = 6.*PI*CHD*SI(1)/(1.+BET)
      DPHI = .5*CL/CHD
      MA = MM/3
      MB = MM-2*((MA+1)/2)
      IF((NT.GT.140).OR.(XP.LT.0.)) JK = -1
      J=1
      DO 40 L = 1,MM
      DELOLD(L) = 0.
      DSUM(J) = 0.
      ARCOLD(L)=ARCL(J)
      IF(J.GE.MM) GO TO 70
      IF((J.LE.MA).OR.(J.GE.MB)) J=J+1
      DSUM(J) = 0.
      J=J+1
   40 CONTINUE
   70 NT = L
      WRITE (N4,100) NT
  100 FORMAT (1H0,I4,45H POINTS WILL BE USED TO DEFINE INNER AIRFOIL )
      CALL SPLIF(MM,ARCL,XC,PHI(1,3),PHI(1,5),PHI(1,7),3,0.,3,0.)
      CALL INTPL(NT,ARCOLD,  XOLD,ARCL,XC,PHI(1,3),PHI(1,5),PHI(1,7))
      CALL SPLIF(MM,ARCL,YC,PHI(1,3),PHI(1,5),PHI(1,7),3,0.,3,0.)
      CALL INTPL(NT,ARCOLD,  YOLD,ARCL,YC,PHI(1,3),PHI(1,5),PHI(1,7))
      CALL SPLIF(MM,ARCL,FM,PHI(1,3),PHI(1,5),PHI(1,7),3,0.,3,0.)
      CALL INTPL(NT,ARCOLD,ANGOLD,ARCL,FM,PHI(1,3),PHI(1,5),PHI(1,7))
      DO 60 L = 1,M
      DO 50 J = 1,NN
   50 PHI(L,J) = R(J)*CO(L)+DPHI*PHIR(L)
   60 CONTINUE
      FSYM = FSYM-12.
      IS = 2
      RETURN
      END
```

```
      SUBROUTINE COSI
C     SET THE SINES,COSINES, AND THE TERM AT INFINITY
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALOI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      TPI = 1./TP
      ANG = ALP+BB(1)
      SN = SIN(ANG)
      CN = SQRT (1.-SN*SN)
      DO 10 L = 1,M
      CO(L) = CN
      SI(L) = SN
      PHIR(L) =(ANG+ATAN((BET*SN*CN)/(1.+BET*SN*SN)))*TPI
      CN = CN*DCN-SN*DSN
      SN = CO(L)*DSN+SN*DCN
      ANG = ANG+DT
   10 CONTINUE
      CO(MM) = CN
      CO(MM+1) = CO(2)
      SI(MM) = SN
      SI(MM+1) = SI(2)
      RETURN
      END


      SUBROUTINE SWEEP
C     SWEEP THROUGH THE GRID ONE TIME
      COMMON/FL/FLUXT4,CD4
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALOI
     5 ,SCALQO,N6,GAMMA,NOPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      YP = 0.
      NSP = 0
      DO 10 J = 1,NN
```

```
      PHI(MM,J) = PHI(1,J)+DPHI
      PHI(MM+1,J) = PHI(2,J)+DPHI
      E(J) = 0.
      RP4(J) = 0.
      RP5(J) = 0.
   10 RPP(J) = 0.
C     SWEEP THROUGH THE GRID FROM NOSE TO TAIL ON UPPER SURFACE
      TE = -2.
      LLP=LL+1
      DO 30 I=LLP,M
      CALL MURMAN
      DO 30 J = 1,N
   30 PHI(I-1,J) = PHI(I-1,J)-RP(J)
      DO 32 J=1,N
   32 PHI(M,J)=PHI(M,J)-E(J)
      DO 51 J=1,N
      E(J)=0.
      RPP(J)=0.
      RP4(J) = 0.
      RP5(J) = 0.
   51 CONTINUE
C     SWEEP THROUGH THE GRID FROM NOSE TO TAIL ON LOWER SURFACE
      TE = 2.
      I = LL
   80 I = I-1
      CALL MURMAN
      DO 60 J = 1,N
   60 PHI(I+1,J) = PHI(I+1,J)-RP(J)
      IF (I.GT.2) GO TO 80
      DO 70 J = 1,N
   70 PHI(2,J) = PHI(2,J)-E(J)
      DO 11 J=1,NN
      PHI(MM+1,J)=PHI(2,J)+DPHI
      E(J)=0.
      RP4(J) = 0.
      RP5(J) = 0.
   11 CONTINUE
      TE=-2.
      I=MM
      CALL MURMAN
      DO 50 J=1,N
      PHI(MM,J)=PHI(MM,J)-E(J)
   50 PHI(1,J)=PHI(MM,J)-DPHI
      DO 12 J=1,N
      E(J)=0.
      RP4(J) = 0.
      RP5(J) = 0.
   12 CONTINUE
      TE=2.
      I=LL
      CALL MURMAN
      DO 13 J=1,N
   13 PHI(LL,J)=PHI(LL,J)-E(J)
C     ADJUST CIRCULATION TO SATISFY THE KUTTA CONDITION
```

```
      IF (RCL .EQ.0.) GO TO 90
      YA = RCL*((PHI(M,1)-(PHI(2,1)+DPHI))*DELTH+SI(1))
      IF (MODE.EQ.1) GO TO 90
      IF (NDES.GE.0) GO TO 41
      ALP = ALP-.5*YA
      GO TO 42
   41 BB(1) = BB(1)-.5*YA
   42 CALL COSI
      GO TO 95
   90 YA = TP*YA/(1.+BET)
      DPHI = DPHI+YA
   95 DO 97 L = 1,M
   97 PHI(L,NN) = DPHI*PHIR(L)
      FLUXT=0.
      NF=N-10
      IF(N.LT.30) NF=N-5
      DO 242 L=2,MM
      U=R(NF)*(PHI(L+1,NF)-PHI(L-1,NF))*DELTH-SI(L)
      V=R(NF)*R(NF)*(PHI(L,NF+1)-PHI(L,NF-1))*DELR -CO(L)
      QF=(U*U+V*V)/FP(L,NF)
      RH=(1.+.2*EM*EM*(1.-QF))**2.5
      FLUX=RH*V/R(NF)
      FLUXT=FLUXT+FLUX
  242 CONTINUE
      FLUXT=DT*FLUXT*CHD
      FLUXT4=FLUXT
      IF(MODE.EQ.0) RETURN
      DO 100 J = 1,N
      DO 100 L = 1,M
  100 PHI(L,J) = PHI(L,J)+YA*PHIR(L)
      RETURN
      END



      SUBROUTINE MURMAN
C     SET UP COEFFICIENT ARRAYS FOR THE TRIDIAGONAL SYSTEM USED FOR LINE
C     RELAXATION AND COMPUTE THE UPDATED PHI ON THIS LINE
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NOPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
C     DO THE BOUNDARY
      E(NN) = 0.
      FAC = -.5*TE
```

```
      IM = I-1
      IF (FAC.LT.0.) IM = I+1
      KK = 0
      PHIO = PHI(I,2)-2.*DR*CO(I)
      PHIYP= PHI(I,2)-PHI(I,1)
      PHIYY = PHIYP+PHIO-PHI(I,1)
      PHIXX = PHI(I+1,1)+PHI(I-1,1)-PHI(I,1)-PHI(I,1)
      PHIXM = PHI(I+1,1)-PHI(I-1,1)
      PHIXP = PHI(I+1,2)-PHI(I-1,2)
C     CHECK FOR THE TAIL POINT
      IF (I.NE.MM) GO TO 10
      C(1) = (C1+C1)*RS(1)
      A(1) = -C(1)+XA*C1-C1
      D(1) = C1*(PHIXX+RS(1)*PHIYY+RA4*CO(I)-E(1))
      GO TO 40
   10 U = PHIXM*DELTH-SI(I)
      BQ = U/FP(I,1)
      QS = U*BQ
      CS = C1-C2*QS
      BQ = BQ*QS*(FP(I-1,1)-FP(I+1,1))
      X = RA4*(CS+QS)*CO(I)
      C(1) = (CS+CS)*RS(1)
      D(1) = CS*RS(1)*PHIYY+RI(1)*BQ+X
      CMQS = CS-QS
      PHIXT = BETA*ABS(U)+ABS(CMQS)
      EM2 = QS/CS
      EPS2 = EPS1*VLAYER(EM2,QPL,QPU)
      PHIXXX= EPS2*PHIXX-RP4(1)
      RP4(1) = EPS2*PHIXX
      D(1) = D(1) + PHIXXX
      IF (QS.LE.QCRIT) GO TO 30
C     FLOW IS SUPERSONIC, BACKWARD DIFFERENCES
      KK = 1
      PHIXT = PHIXT-CMQS
      PHIXXM = RPP(1)
      A(1) = -(C(1)+PHIXT)
      D(1) = D(1)+CMQS*PHIXXM-PHIXT*E(1)
      A(1) = A(1)-2.*EPS2-RP5(1)
      D(1) = D(1) -(EPS2+2.*RP5(1))*E(1)
      RP5(1) = EPS2
      GO TO 40
C     FLOW SUBCRITICAL, CENTRAL DIFFERENCES
   30 A(1) = XA*CMQS -C(1)-PHIXT
      D(1) = D(1)+CMQS*PHIXX-PHIXT*E(1)
      A(1) = A(1)-2.*EPS2-RP5(1)
      D(1) = D(1) -(EPS2+2.*RP5(1))*E(1)
      RP5(1) = EPS2
C     DO NON-BOUNDARY POINTS
   40 RPP(1) = PHIXX
      DO 60 J = 2,N
      PHIXX = PHI(I+1,J)+PHI(I-1,J)-PHI(I,J)-PHI(I,J)
      DU = PHIXP
      PHIXP = PHI(I+1,J+1)-PHI(I-1,J+1)
      PHIXY = PHIXP-PHIXM+(E(J+1)-E(J-1))*FAC
```

```
      PHIXM = DU
      DU = DU*DELTH
      PHIYYM = PHIYY
      PHIYM = PHIYP
      PHIYP = PHI(I,J+1)-PHI(I,J)
      PHIYY = PHIYP-PHIYM
      U = R(J)*DU-SI(I)
      DV = R(J)*(PHI(I,J+1)-PHI(I,J-1))*DELR
      V = DV*R(J)-CO(I)
      RAV = R(J)*RA*V
      BQ = 1./FP(I,J)
      BQU = BQ*U
      US = BQU*U
      UV = (BQU+BQU)*V
      VS = BQ*V*V
      QS = US+VS
      CS = C1-C2*QS
      CMVS = CS-VS
      CMUS = CS-US
      PHIXT = BETA*ABS(U)
      PHIYT = BETA*ABS(RAV)
      EM2 = QS/CS
      EPS2 = EPS1*VLAYER(EM2,QPL,QPU)
      PHIXXX= EPS2*PHIXX-RP4(J)
      RP4(J) = EPS2*PHIXX
C     COMPUTE CONTRIBUTION OF RIGHT-HAND SIDE FROM LOW ORDER TERMS
      D(J) =RA4*((CMVS+US-VS)*DV-UV*DU)+RI(J)*QS*BQ*(U*(FP(I-1,J)-
     1 FP(I+1,J))+RAV*(FP(I,J-1)-FP(I,J+1)))
      D(J) = D(J) +PHIXXX
      UV = .5*BQU*RAV
      IF (QS.LE.QCRIT) GO TO 50
C     SUPERSONIC FLOW, USE BACKWARD DIFFERENCING
      KK = KK+1
      CMQS = CS-QS
      FQ = 1./QS
      AUU = US*FQ
      BUU = RS(J)*AUU
      BVV = VS*FQ
      AVV = RS(J)*BVV
      BUV = UV*FQ
      AUV = BQU*ABS(RAV)*FQ*TE
      PHINN = BVV*PHIXX-BUV*PHIXY+BUU*PHIYY
      B(J) =  CS*BUU
      PHIXT = PHIXT-CMQS*(AUU+AUU-AUV) +CS*BVV
      PHIYT = PHIYT -CMQS*(AVV+AVV-AUV)
      C(J) = B(J)+PHIYT
      PHIXXM = RPP(J)
      IF (V.LT.0) GO TO 45
      PHIYYM = PHI(I,J+2)-PHI(I,J+1)-PHIYP
      PHIXYM = PHIYP+PHI(IM,J)-PHI(IM,J+1)
      GO TO 46
   45 PHIXYM = PHI(IM,J)-PHI(IM,J-1)-PHIYM
      BQ = B(J)
      B(J) = C(J)
```

```
      C(J) = BO
   46 PHISS = AUU*PHIXXM+AUV*PHIXYM+AVV*PHIYYM
      A(J) = -(B(J)+C(J)+PHIXT)
      D(J) = D(J)+CMUS*PHISS+CS*PHINN-E(J)*PHIXT
      A(J) = A(J)-2.*EPS2-RP5(J)
      D(J) = D(J) -(EPS2+2.*RP5(J))*E(J)
      RP5(J) = EPS2
      GO TO 60
C     SUBSONIC FLOW, USE CENTRAL DIFFERENCES
   50 C(J) =  RS(J)*CMVS
      B(J) = C(J)+PHIYT
      PHIXT = PHIXT+CMUS
      A(J) =  XA*CMUS-B(J)-C(J)-PHIXT
      D(J) = D(J)+CMUS*PHIXX-UV*PHIXY+C(J)*PHIYY-PHIXT*E(J)
      A(J) = A(J)-2.*EPS2-RP5(J)
      D(J) = D(J) -(EPS2+2.*RP5(J))*E(J)
      RP5(J) = EPS2
      IF (V.LT.0.) GO TO 60
      B(J) = C(J)
      C(J) = C(J)+PHIYT
   60 RPP(J) = PHIXX
      NSP = NSP+KK
C     SOLVE THE TRIDIAGONAL SYSTEM
      CALL TRID
      RETURN
      END



      SUBROUTINE TRID
C     SOLVE N DIMENSIONAL TRIDIAGONAL SYSTEM OF EQUATIONS
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),F(31),RS(31),R1(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),CSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DP,DELTH,DELR,PA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDLS,XLEN,SCALOI
     5 ,SCALOO,N6,GAMMA,NOPT,CSTAR,FFM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,FLTSZ,QPL,QPU
      XX = 1./A(1)
      RP(1) = E(1)
      E(1) = XX*D(1)
C     DO ELIMINATION
      DO 10 J = 2,N
      C(J-1) = C(J-1)*XX
      XX = 1./(A(J)-B(J)*C(J-1))
      RP(J) = E(J)
   10 E(J) = (D(J)-B(J)*E(J-1))*XX
C     DO BACK SUBSTITUTION
```

```
      EMX = ABS(E(N))
      DO 20 J = 2,N
      L = NN-J
      E(L) = E(L)-C(L)*E(L+1)
   20 EMX = AMAX1(EMX,ABS(E(L)))
C     FIND THE LOCATION OF THE MAXIMUM RESIDUAL
      IF (EMX.LE.ABS(YR)) RETURN
      IK = I
      DO 70 J = 1,N
      IF (ABS(E(J)).EQ.EMX) GO TO 74
   70 CONTINUE
   74 JK = J
      YR = E(JK)
      RETURN
      END



      SUBROUTINE REMESH(LSIGN)
C     GO TO CRUDER GRID IF LSIGN IS -1
C     GO TO FINER GRID IF LSIGN IS +1
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162)-PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      X = 2.**LSIGN
      NG = FLOAT(NG)/X+.2
      M = FLOAT(M)*X +.2
      N = FLOAT(N)*X+.2
      IF (N.EQ.14) N=15
      LL = FLOAT(LL-1)*X+1.2
      IF (LSIGN.GT.0) MM = M+1
      IF (LSIGN.GT.0) NN = N+1
      LSEP = FLOAT(LSEP-1)*X+1.2
      PF = 1./X
      DELR = X*DELR
      DELTH = X*DELTH
      DR = PF*DR
      DT = PF*DT
      DCN = COS(DT)
      DSN = SIN(DT)
      RA4 = PF*PF*RA4
      NCY = 0
      I = LSIGN
      MP = MM+1
```

```
      CALL PERMUT (R,NN,1)
      CALL PERMUT (RS,NN,1)
      DO 5 J = 1,N
    5 RI(J) = -.25*DT/R(J)
      CALL PERMUT (DSUM,MP,1)
      DO 20 L = 1,NN
   20 CALL PERMUT (PHI(1,L),MP,1)
      DO 30 L = 1,MP
   30 CALL PERMUT (PHI(L,1),NN,IDIM)
      MM = M+1
      NN = N+1
      IF (X.EQ..5) GO TO 80
      DO 40 L = 1,M,2
      DSUM(L+1) = .5*(DSUM(L)+DSUM(L+2))
      DO 40 J = 1,NN,2
   40 PHI(L+1,J) = .5*(PHI(L,J)+PHI(L+2,J))
      DO 50 J = 1,N,2
      DO 50 L = 1,MM
   50 PHI(L,J+1) = .5*(PHI(L,J)+PHI(L,J+2))
   80 CALL MAP
      RETURN
      END



      SUBROUTINE PERMUT (AX,NX,JX)
C     REORDERS POINTS WITHIN AN ARRAY
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CC(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,FN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,WCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MOLE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IYX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NLES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,FEM,DEP,WINF,TSTEP,XLUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,UPL,QPU
      DIMENSION AX(1)
      L = 1
      JY = JX+JX
      NY = 2*((NX-1)/2)+1
      NZ = 2*(NX/2)
      IF(I.GT.0) GO TO 30
      NY = JX*(NY-1)+1
      NZ = JX*(NZ-1)
      DO 10 J = 1,NY,JY
      A(L) = AX(J)
   10 L = L+1
      DO 20 J = JX,NZ,JY
      A(L) = AX(J+1)
   20 L = L+1
```

```
      GO TO 60
   30 DO 40 J = 1,NY,2
      A(J) = AX(L)
   40 L = L+JX
      DO 50 J = 2,NZ,2
      A(J) = AX(L)
   50 L = L+JX
   60 L = 1
      DO 70 J = 1,NX
      AX(L) = A(J)
   70 L = L+JX
      RETURN
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      SUBROUTINE GETCP(CDF)
C     COMPUTE CP,CD, AND CM BY INTEGRATION AND OUTPUT MACH DIAGRAM
      COMMON/FL/FLUXT4,CD4,CDW,INDCD
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),F(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CC(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),APCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),AKCOLD(162),DELOLD(162)
     4 ,PP4(31),RP5(31)
      COMMON /A/ PI,TF,RAD,E1,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BEI,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NE4,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,NO,GAMMA,NOPT,CSTAR,REM,DEP,WINF,TOTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      REAL MACHN,MACH
      COMPLEX CLCD,TMP
      DIMENSION MACHN(1),CPX(1),MN(1),IMACH(21)
      EQUIVALENCE (MACHN(1),A(1)),(CPX(1),PHIR(1)),(MN(1),FP(1,31))
      DATA IMACH/1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ,1H0,1H1,1H2,1H3
     1,1H4,1H5,1H6,1H7,1H8,1H9,1H+/
      DATA TX /4HCDF=/
      MACH(Q) = SQRT(Q/(C1-C2*Q))
      IMC(Q) = MIN0(21,IFIX(10.*Q)+1)
      CLCD = 0.
      CM = 0.
      IF ((XP.GT.0.).OR.(IZ.LE.30)) GO TO 10
      DY = YOLD(NT)-YOLD(1)
      IF (FSYM.NE.0.) DY = YC(MM)-YC(1)
      REWIND M4
      WRITE (M4,120) EM,CL,DY,TC, NRN,MM
   10 DO 20 L = 1,MM
      CP = CPX(L)
C     COMPUTE CP*DZ
      TMP =  CP*SQRT(FP(L,1))*CMPLX(COS(FM(L)),SIN(FM(L)))
C     SUM UP CL,CD, AND CM
      CLCD = CLCD+TMP
```

```
      CM = CM+(XC(L)-.25)*REAL(TMP)-YC(L)*AIMAG(TMP)
C     WRITE PUNCH OUTPUT ON M4 IF XP=0 AND IZ.GT.80
      IF ((XP.GT.0.).OR.(IZ.LE.80)) GO TO 20
      Q = MACHN(L)*SQRT(C1/(1.+C2*MACHN(L)*MACHN(L)))
      V = Q*SIN(FM(L))
      U = Q*COS(FM(L))
      IF (XP.EQ.0) GO TO 15
      WRITE (M4,130) U,V,XC(L),YOLD(L),CP
      GO TO 20
   15 WRITE (M4,130) U,V,XC(L),YC(L),CP
   20 CONTINUE
C     CORRECT CL,CD FOR ANGLE OF ATTACK
      CLCD = -(D T*CHD)*CLCD*CMPLX(SIN(ALP),COS(ALP))
      CM = DT*CHD*CM
C     WRITE CD,CL,CM ONTO N4
      CDW = REAL(CLCD)
      QCR=SQRT(QCRIT)
      DCD4=2.*(QCR-1.)*FLUXT4
      CD = CDW+CDF
      CD4=CD+DCD4
      CDW=CDW+DCD4
      IF(INDCD.EQ.0) PRINT 261,CDW,CDF,CD4
  261 FORMAT(5H CDW=F10.5,5H CDF=F10.5,4H CD=F10.5)
      CL2 = AIMAG(CLCD)
      IF (INDCD.EQ.0) GO TO 160
      CALL COSI
      RETURN
  160 CONTINUE
      IF (M4.EQ.N3) GO TO 85
      IF (CDF.EQ.0.) GO TO 70
      WRITE (N4,90) EM,CL2,CM,CDW,TX,CDF,CD4
      GO TO 80
   70 WRITE (N4,90) EM,CL2,CM,CD4
C     CONSTRUCT MACH NUMBER DIAGRAM
      WRITE (N4,140)
   80 I = IMC(EM)
      I = IMACH(I)
C     USE PRINT WIDTH OF IZ FOR MACH NUMBER DIAGRAM
      MB = MM
      MC = MAX0(1,MB/IZ)
      MA = MC+MAX0(1,MB-IZ*MC)
C     WRITE OUT MACH NUMBERS AT INFINITY
      WRITE (N4,100) (I, L = MA,MB,MC)
C     DO MACH NUMBERS ONE LINE AT A TIME DOWN TO THE BODY
      J = NN-MC
   40 RSJ = R(J)*R(J)
      DO 50 L = MA,MB,MC
      U = (PHI(L+1,J)-PHI(L-1,J))*R(J)*DELTH-SI(L)
      V = (PHI(L,J+1)-PHI(L,J-1))*DELR*RSJ  -CO(L)
      Q = (U*U+V*V)/FF(L,J)
      I = IMC(MACH(Q))
      MN(L) = IMACH(I)
   50 CONTINUE
      WRITE (N4,100) (MN(L),L = MA,MB,MC)
```

```
      J = J-MC
      IF (J.GT.1) GO TO 40
C     DO THE LINE WHICH IS THE BODY
      DO 60 L = MA,MB,MC
      I = IMC(MACHN(L))
   60 MN(L) = IMACH(I)
      WRITE (N4,100) (MN(L),L = MA,MB,MC)
      IF (ITYP.GE.4) CALL GRAFIC(CD)
      RETURN
   85 RNX = .1*AINT(RN*1.E-5)
      WRITE (N4,150) EM,CL,TC,CM,RNX,CDF
      RETURN
   90 FORMAT (1H12X3HEM=F5.4,4X3HCL=F7.4,4X3HCM=F6.4,4X4HCDW=F7.5,4XA4
     1 ,F7.5,4X 3HCD=F7.5///)
  100 FORMAT (3X,130A1)
  120 FORMAT (3H M=,F4.3,5X,3HCL=,F5.3,5X,3HCY=,F5.3,6X,4HT/C=,
     1 F4.3,14X,2I5)
  130 FORMAT (40Z0)
  140 FORMAT (1H0//)
  150 FORMAT (1H0//7X3HEM=,F4.3,4X3HCL=,F6.4,4X4HT/C=,F4.3,4X3HCM=,
     1 F6.4,4X3HRN=,F4.1,4X4HCDF=,F6.4/)
      END



      SUBROUTINE GRAFIC(CD)
      COMPLEX ZP,ZQ,SFAC,SIG
      REAL MACHN
      COMMON/FL/FLUXT4,CD4,CDW,INCCD
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),PS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TITLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NI,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALO^-N6,GAMMA,NWPT,CSTAR,REM,DEP,WINF,TSTEP,XCUT
     6 ,INC,    C,GAM,KDES,PLTSZ,QPL,QPU
      DIMENSION CPX(1),MACHN(1),T(6)
      EQUIVALENCE (CPX(1),PHIR(1)) ,(MACHN(1),A(1))
      DATA TOL/1.E-6/ , PF/-.4/ , SCF/5.0/,YCR/4.0/,SIZE/.14/,SCD/200./
C     MOVE THE ORIGIN TWO INCHES OVER AND TWO INCHES UP
      CALL PLOT(2.0,2.5,-3)
      YCR = AMAX1(3.5,.5*AINT(20.*EM-7.0))
C     PLOT CP CURVE AS A FUNCTION OF X
      CPF = 1./PF
      CCP = CPF*CPX(1)
      CALL PLOT(SCF*XC(1),YCR+CCP,3)
      DO 10 L = 2,MM
      CCP = AMIN1(8.5-YCR,CPF*CPX(L))
```

```
   10 CALL PLOT(SCF*XC(L),YOR+CCP,2)
C     DRAW AND LABEL THE CP-AXIS
      CALL CPAXIS(-.5,YOR,1.-1./PF,7.5-YOR,PF)
C     COMPUTE AND PLOT CRITICAL SPEED
      CALL SYMBOL (-.5,YOR+CPF*CPX(MM+1),2.*SIZE,15,0.,-1)
C     PLOT BODY
      CALL PLOT(SCF*XC(1),SCF*YC(1),3)
      DO 20 L = 2,MM
   20 CALL PLOT(SCF*XC(L),SCF*YC(L),2)
C     LABEL THE PLOT
      ALPX = RAD*ALP
      TXT=8HANALYSIS
      IF (EPS1.GT.0.) TXT = 10HART. VISC.
      IF(FSYM.GE.6.) TXT=6HTHEORY
      XL=-.9
C     ****NON-ANSI - SEE WRITEUP AT END****
      IF(FSYM.GE.6.) GO TO 30
      IF ((NDES.LT.0).AND.(EPS1.LE.0.)) GO TO 200
      TTLE(1) = 4HVISC
      TTLE(2) = 4HOUS
      TTLE(3) = 4HDESI
      TTLE(4) = 4HGN
      ENCODE (60,210,T) TTLE,M,N,NCY,EPS1
      GO TO 40
  200 ENCODE (60,191,T) TTLE,M,N,NCY
      GO TO 40
   30 LN=RN*1.E-6+.5
      ENCODE(60,190,T) TTLE,M,N,NCY,LN
   40 CALL SYMBOL(-1.14,-1.0,SIZE,T,0.,56)
C     ****NON-ANSI - SEE WRITEUP AT END****
      ENCODE(60,170,T) TXT,EM,ALPX,CL,CD4
      IF(CD4.LT.0) ENCODE(60,171,T) TXT,EM,ALPX,CL,CD4
      CALL SYMBOL(XL,-1.35,SIZE,T,0.,60)
      CALL SYMBOL(XL-.10,-1.35+.5*SIZE,1.5*SIZE,15,0.,-1)
      CN=CO(1)
      SN=SI(1)
C     READ AND PLOT EXPERIMENTAL DATA IF XP IS NOT ZERO
      IF (XP.EQ.0.) GO TO 130
      REWIND M4
      READ (M4,140) NP
      IF (EOF(M4).NE.0) GO TO 130
      IF (NDES.GE.0) GO TO 220
      READ (M4,150) EMX,ALPX,CLX,CDX,SNX
      READ (M4,160) (CO(L),SI(L),L = 1,NP)
      TXT = 10HEXPERIMENT
      GO TO 230
  220 READ (M4,240)       TCX,DQAVE,YRX,SNX
      READ (M4,160) (CO(L),SI(L),L = 1,NP)
      TXT =  8HINPUT CP
  230 CONTINUE
      NC=59
      IF(SNX.GE.0.)GO TO 50
      TXT=6HDESIGN
      NC=3
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C      ****NON-ANSI - SEE WRITEUP AT END****
   50 ENCODE (60,170,T) TXT,EMX,ALPX,CLX,CDX
      IF (CD4.LT.0) ENCODE(60, 71,T) TXT,EMX,ALPX,CLX,CDX
      IF(NDES.GE.0) ENCODE(60,250,T) TXT,TCX,DQAVE,YRX
      CALL SYMBOL(XL,-1.7,SIZE,T,0.,60)
      CALL SYMBOL(XL-.10,-1.7+.5*SIZE,SIZE,NC,0.,-1)
      DO 180 L = 1,NP
      CCP = YOR+CPF*SI(L)
      IF (CCP.GT.8.4) GO TO 180
      CALL SYMBOL(SCF*CO(L),CCP,.5*SIZE,NC,0.,-1)
  180 CONTINUE
  130 IF (ITYP.EQ.5) GO TO 122
C     PLOT THE SONIC LINE
      EX = 1.-EPSIL
C     SET SINES AND COSINES FOR USE IN FOURIER SERIES
      MX = M/2
      CO(1) = 1.
      SI(1) = 0.
      DO 60 L = 1,MX
      CO(L+1) = CO(L)*DCN-SI(L)*DSN
      CO(MM-L) = CO(L+1)
      SI(L+1) = CO(L)*DSN+SI(L)*DCN
   60 SI(MM-L) = -SI(L+1)
      DO 120 L = 2,M
C     LOOK FOR SONIC POINTS ON THE BODY
      IF (MACHN(L).LT.1.) GO TO 110
      IF (MACHN(L-1).GE.1.) GO TO 80
      IPEN = 3
C     COMPUTE Z AT SONIC LINE ON BODY
   70 R1 = (MACHN(L)-1.)/(MACHN(L)-MACHN(L-1))
      ZP = CMPLX(XC(L)+R1*(XC(L-1)-XC(L)),YC(L)+R1*(YC(L-1)-YC(L)))
      CALL PLOT(SCF*REAL(ZP),SCF*AIMAG(ZP),IPEN)
      IF (IPEN.EQ.2) GO TO 120
C     FIND THE SONIC LINE ALONG A RAY
   80 Q = MACHN(L)
      SX = SI(L)*CN+SN*CO(L)
      CX = CO(L)*CN-SN*SI(L)
      FAC = .5*DR
      ZQ = CMPLX(XC(L),YC(L))
      DO 90 J = 1,N
      ZP = SFAC
      RJ = R(J)
      QS = Q
      IF (J.EQ.1) GO TO 82
      U = (PHI(L+1,J)-PHI(L-1,J))*RJ*DELTH-SX
      V = (PHI(L,J+1)-PHI(L,J-1))*DELR*RJ*RJ-CX
      Q = (U*U+V*V)/FP(L,J)
      Q = SQRT(Q/(C1-C2*Q))
   82 SIG = CMPLX(RJ*CO(L),RJ*SI(L))
C     COMPUTE ((1-SIGMA)**(1-EPSIL))SIGMA
      SFAC = CEXP(EX*CLOG((1.,0.)-SIG))/SIG
C     SUM UP FOURIER SERIES TO OBTAIN CONJUGATE OF W
      S = -BB(1)
      DO 84 K = 1,NFC
```

```
      LT = MOD((L-1)*K,M)
      S = S+RJ*(AA(K+1)*SI(LT+1)-BB(K+1)*CO(LT+1))
      RJ = RJ*R(J)
      IF (RJ.LT.TOL) GO TO 86
   84 CONTINUE
C     COMPUTE THE ARGUMENT OF DZ/DR
   86 SFAC = -SFAC*CMPLX(COS(S),SIN(S))/CABS(SFAC)
C     MULTIPLY THE ARGUMENT BY THE MAGNITUDE TO OBTAIN DZ/DR
      SFAC = SFAC*(CHD*SQRT(FP(L,J)))/(R(J)*R(J))
C     PERFORM THE INTEGRATION
      ZQ = ZQ+FAC*SFAC
      FAC = DR
      IF (Q.LE.1.) GO TO 100
   90 CONTINUE
  100 ZQ = ZQ-.5*DR*SFAC
      ZP = ZQ-.5*DR*(SFAC+ZP)
      R1 = (Q-1.)/(Q-QS)
      ZP = ZQ+R1*(ZP-ZQ)
      CALL PLOT (SCF*REAL(ZP),AMAX1(-2.0,SCF*AIMAG(ZP)),2)
      GO TO 120
  110 IPEN = 2
      IF (MACHN(L-1).GE.1.) GO TO 70
  120 CONTINUE
C     POSITION PEN AT BEGINNING OF NEXT PAGE
  122 CALL FRAME
      CALL PLOT(-2.0,-2.5,-3)
      IF ((FSYM.NE.7.).OR.(ITYP.EQ.6)) RETURN
C     PLOT THE BOUNDARY LAYER DISPLACEMENT
      MX = INDEXR (0.,XC,M)
      CALL PLOT(2.,1.5,-3)
      CALL SYMBOL(1.36,-.65,SIZE,19HLOWER SURFACE  DELS ,0.,19)
      CALL CPAXIS (0.,0.,0.,4.,1./SCD)
C     PLOT LOWER SURFACE
      CALL PLOT (SCF*XC(1),SCD*DSUM(1),3)
      DO 132 L = 2,MX
  132 CALL PLOT (SCF*XC(L),SCD*DSUM(L),2)
      CALL PLOT(0.,4.5,-3)
      CALL SYMBOL(1.36,-.65,SIZE,19HUPPER SURFACE  DELS ,0.,19)
      CALL CPAXIS (0.,0.,0.,4.,1./SCD)
C     PLOT UPPER SURFACE
      CALL PLOT (SCF*XC(MX),SCD*DSUM(MX),3)
      DO 134 L = MX,M
  134 CALL PLOT (SCF*XC(L+1),SCD*DSUM(L+1),2)
      CALL PLOT(10.,-6.,-3)
      RETURN
  140 FORMAT (10X,I3)
  150 FORMAT (3F6.3,F7.5,E9.1)
  160 FORMAT (2F10.4)
  170 FORMAT (A12,4H  M=F4.3,3X4HALP=F5.2,3X3HCL=,F5.3,3X3HCD=,F5.4)
  171 FORMAT(A12,4H  M=F4.3,3X4HALP=F5.2,3X3HCL=,F5.3,2X3HCD=F6.4)
  190 FORMAT(4A4,3X4HM*N=I3,1H*I2,3X4HNCY=I4,4X2HR=I2,8H MILLION)
  191 FORMAT(4A4,3X4HM*N=I3,1H*I2,3X4HNCY=I4,4X12HNO VISCOSITY)
  240 FORMAT  (F7.3,2E10.2,F4.1)
  210 FORMAT(4A4,3X4HM*N=I3,1H*I2,3X4HNCY=I4,4X,5HEPS1=,F5.3)
```

```
  250 FORMAT(A12,2X,4HT/C=F5.3,2X,3HDQ=E8.2,2X,5HDPHI=E8.2)
      END



      SUBROUTINE CPAXIS(XOR,YOR,BOT,TOP,SCF)
C     DRAWS AND LABELS THE CP AXIS
C     XOR,YOR IS THE LOCATION OF THE ORIGIN OF THE AXIS
C     BOT IS THE LENGTH OF THE AXIS BELOW THE ORIGIN
C     SCF IS A SCALE FACTOR USED FOR LABELING
C     DRAW THE LINE FOR THE AXIS
C     SCF NEGATIVE FOR CP AXIS AND POSITIVE FOR DELS AXIS
      SIZE = .12-SIGN(.02,SCF)
      CALL PLOT (XOR,YOR+TOP,3)
      CALL PLOT (XOR,YOR-BOT,2)
C     DRAW HATCH MARKS AND LABELS ONE INCH APART
      N = 1+INT(BOT)+INT(TOP)
      S = -AINT(BOT)*SCF +1.E-12
      XH = XOR-(3.*SIZE)/.7
      YH = YOR-AINT(BOT)
      DO 10 I = 1,N
      CALL SYMBOL (XOR,YH,SIZE,15,0.,-1)
C     ****NON-ANSI -  SEE WRITEUP AT THE END****
      IF (SCF.GT.0.) ENCODE (10,25,A) S
      IF (SCF.LE.0.) ENCODE (10,20,A) S
      S = S+SCF
      CALL SYMBOL (XH,YH,SIZE,A,0.,4)
   10 YH = YH+1.
      IF (SCF.GT.0.) GO TO 30
      CALL SYMBOL(XOR+.1,YOR+2.5,.14,1HC,0.,1)
      CALL SYMBOL(XOR+.25,YOR+2.38,.14,1HP,0.,1)
      RETURN
C     DRAW THE X-AXIS
   30 CALL PLOT (XOR,YOR-BOT,3)
      CALL PLOT (XOR+5.0,YOR-BOT,2)
      CALL SYMBOL (XOR+5.5,YOR-.07,.14,1HX,0.,1)
      YH = YOR-BOT-SIZE-SIZE
      DO 40 I = 1,5
      S = .2*FLOAT(I)
      ENCODE (10,20,A) S
      XH = YOR+FLOAT(I)-SIZE-SIZE
      CALL SYMBOL (XH,YH,SIZE,A,0.,4)
   40 CALL SYMBOL (XOR+FLOAT(I),YOR-BOT,SIZE,15,90.,-1)
      CALL SYMBOL (XOR+.25,YOR+3.0,.14,4HDELS,0.,4)
      RETURN
   25 FORMAT ( F4.3)
   20 FORMAT (F4.1)
      END
```

```
      SUBROUTINE GOPLOT (NRN)
C     INITIATE PLOT
C     *****************************************************************
C     THIS SUBROUTINE SHOULD BE REPLACED BY ANY ROUTINE WHICH INSTRUCTS
C     THE SYSTEM TO INITIATE A PLOT
C     *****************************************************************
      IF (NRN.GT.1000) GO TO 50
      CALL PLOTS(600,26HJEFF MCFADDEN, FOLDER 3210)
      RETURN
   50 CALL PLOTSBL(600,26HJEFF MCFADDEN, FOLDER 3210)
      RETURN
      END




      SUBROUTINE AIRFOL
C     READS IN DATA FOR AIRFOIL AND MAKES INITIAL GUESS FOR MAPPING
C     FUNCTION BY COMPUTING FOURIER COEFFICIENTS
C     IF ONLY X,Y COORDINATES ARE PRESCRIBED SMOOTHING IS DONE
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,KN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,CCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3 N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      DIMENSION XX(1),YY(1),U(1),V(1),W(1),SP(1),CIRC(1),TH(1),TT(1)
     1 ,DS(1),SS(1),CX(1),SX(1),QSP(1),TITLE(15),Z(1)
      EQUIVALENCE(XX(1),FP(1,3 )),(YY(1),FP(1,5)),(U(1),FP(1,1)),
     1 (V(1),FP(1,7)),(W(1),FP(1,9)),(SP(1),FP(1,11)),(CIRC(1),FP
     2 (1,13)),(TH(1),FP(1,15)),(TT(1),FP(1,17)),(DS(1),FP(1,19)),
     3 (SS(1),FP(1,21)),(CX(1),FP(1,23)),(SX(1),FP(1,25)),(QSR(1),
     4 FP(1,27)),(Z(1),FP(1,29))
      SQ(Q2) = Q2*Q2
      SMOOTH(Q1,Q2,Q3,Q4) = Q2+SQ(SQ(SQ(Q4)))*.25*(Q1-Q2-Q2+Q3)
      DIS(Q1) = (Q1-ERR)*((Q1-ERR)*(Q1-ERR)+CONST)
      DATA TOL,NT,ISYM,CONST,VAL/.4E-7,999,0,.2,4HRUN /
      DATA DXDS1,DXDS2,DYDS1,DYDS2/4*0./ , XT/-1./
C     NMP IS THE NUMBER OF POINTS IN CIRCLE PLANE FOR FOURIER SERIES
      LC = NFC
      NMP = 2*LC
      MC = NMP + 1
      PILC = PI/FLOAT(LC)
      IF (FSYM.GE.6.) GO TO 150
      WRITE (N4,470)
      REWIND N3
      READ (N3,410) TITLE
      IF (FSYM.GE.3.) GO TO 100
```

```
C     READ IN COORDINATES AS PRODUCED BY PROGRAMS D AND F
      EPSIL = 2.
      XX(1) = 0.
      NL = 2
      REWIND N3
      READ (N3,510) EM,CL,DY,TC,NRN
      IMC = MOD(INT(100.*EM+.5),100)
      ICL1 = MOD(INT(CL+.05),10)
      ICL2 = MOD(INT(10.*CL+.5),10)
      ITC1 = MOD(INT(10.*TC+.05),10)
      ITC2 = MOD(INT(100.*TC+.5),10)
      ENCODE (40,530,TTLE) IMC,ICL1,ICL2,ITC1,ITC2
      MODE = 0
      IF (NRN.LT.0) FSYM=2.
      DO 40 L = 1,999
      READ (N3,500) U(L),V(L),XX(L),YY(L),FAC
C     ****CHECK FOR END OF FILE****
      IF (EOF(N3).NE.0) GO TO 50
      IF (XX(L).LT.XX(NL)) NL = L
   40 CONTINUE
C     AIRFOIL HAS BEEN EXTENDED IN PROGRAM D
   50 NT = L-1
      NRN = IABS(NRN)
      GO TO 150
C     READ IN AIRFOIL DATA FROM CARDS
  100 READ (N3,420) FNU,FNL,EPSIL
      READ (N3,470)
      NT = FNU+FNL-1.
      NL = FNL
      DO 110 I = NL,NT
  110 READ (N3,420) U(I),V(I),XX(I),YY(I)
      READ (N3,470)
      DO 120 I = 1,NL
      J = NL+1-I
  120 READ (N3,420) U(J),V(J),XX(J),YY(J)
      DO 130 J = 1,4
  130 TTLE(J) = TITLE(J)
      IF (FSYM.LE.4.) GO TO 150
      DO 140 L = 1,NT
      TH(L) = XX(L)/RAD
      XX(L) = U(L)
  140 YY(L) = V(L)
      GO TO 195
C     NO PERIOD IN THE STREAM FUNCTION
   95 EPSIL = 0.
C     DEFINE SLOPES SO THAT ARC LENGTHS CAN BE COMPUTED TO FIRST ORDER
  150 IF ((FSYM.EQ.1.).OR.(FSYM.EQ.3.)) GO TO 170
      DO 160 I = 1,NT
  160 TH(I) = 0.
      ISYM = 1
      GO TO 200
C     COMPUTE SLOPES FROM VELOCITIES
  170 TH(1) = ATAN(V(1)/U(1))
      QSR(1) = U(1)*U(1)+V(1)*V(1)
```

```
      DO 190 I = 2,NT
C     CHOOSE NEAREST BRANCH FOR THE ARCTANGENT
      DTH = ATAN((U(I-1)*V(I)-U(I)*V(I-1))/(U(I-1)*U(I)+V(I-1)*V(I)))
      TH(I) = TH(I-1)+DTH
  190 QSR(I) = U(I)*U(I)+V(I)*V(I)
  195 IF (EPSIL.GT.1.) EPSIL = (TH(1)-(PI+TH(NT)))/PI
      IF (FSYM.GT.5.) EPSIL = (TH(1)+TH(2)-TH(NT)-TH(NT-1))/TP-1.
C     COMPUTE ARC LENGTH TO FOURTH ORDER ACCURACY
  200 SP(1) = 0.
      DO 210 I = 2,NT
      DUM = AMAX1(.1E-20,.5*ABS(TH(I)-TH(I-1)))
      DX = XX(I)-XX(I-1)
      DY = YY(I)-YY(I-1)
  210 SP(I) = SP(I-1)+SQRT(DX*DX+DY*DY)*DUM/SIN(DUM)
      ARC = SP(NT)
      SN = 2./ARC
      SCALE = .25*ARC
      EE = .5*(1.-EPSIL)
      DO 220 L = 1,NT
  220 SS(L) = ACOS(1.-SN*SP(L))
      SS(NT) = PI
      IF (ISYM.NE.0) GO TO 350
      CALL SPLIF (NT,SS,TH,U,V,W,3,0.,3,0.)
      IF (FSYM.GT.5.) GO TO 232
      WRITE (N4,410) TITLE,VAL,NRN
      IF (N4.NE.N2) WRITE (N2,410) TITLE,VAL,NRN
C     PRINT OUT DATA ON THE AIRFOIL
      WRITE (N4,430)
      DO 230 L = 1,NT
      VAL = TH(L)*RAD
      SUM =-SN*U(L)/AMAX1(.1E-5,SIN(SS(L)))
      IF ((L.EQ.1).OR.(L.EQ.NT)) SUM = V(L)*SIGN(SN,FLOAT(L-2))
  230 WRITE (N4,480) XX(L),YY(L),SP(L),VAL,SUM,V(L),W(L)
      WRITE (N4,440)
C     MAKE INITIAL GUESS OF ARC LENGTH AS A FUNCTION OF CIRCLE ANGLE
  232 DX = (XX(NT)-XX(1))/TP
      DY = (YY(NT)-YY(1))/TP
      DO 240 I = 1,MC
      ANGL = FLOAT(I-1)*PILC
      CIRC(I) = ANGL
      CX(I) = COS(ANGL)
      SX(I) = SIN(ANGL)
      YY(I) = 1.
      IF (EE.NE.0.) YY(I) = (2.-2.*CX(I))**EE
      FAC = SIGN(1.+CX(I),FLOAT(LC-I))
  240 SP(I) = ACOS(.5*FAC)
      SP(MC) = PI
      CIRC(MC) = TP
      IF (FSYM.LT.6.) GO TO 244
      SCALE = ARC/ARCL(MM)
      SN1=2./ARCL(MM)
      DO 322 I=1,M
  322 ARCL(I)=ACOS(1.-SN1*ARCL(I))
      ARCL(MM)=PI
```

```
      DO 242 L = 1,MM
  242 Z(L) = FLOAT(L-1)*DT
      CALL SPLIF (MM,Z,ARCL,CO,SI,PHIR,3,0.,3,0.)
      CALL INTPL (NMP,CIFC,SP,Z,ARCL,CO,SI,PHIR)
  244 DO 245 L = 1,LC
      BB(L) = CX(2*L-1)
  245 AA(L) = -SX(2*L-1)
C     DO AT MOST 100 ITERATIONS TO FIND THE FOURIER COEFFICIENTS
      DO 320 K   = 1,100
      CALL INTPL(NMP,SP,TT,SS,TH,U,V,W)
      DO 250 I = 1,NMP
  250 TT(I) = TT(I)+.5*(CIRC(I)+EPSIL*(CIRC(I)-PI))
      TT(1)=.5*(TH(1)+TH(NT)+PI)
C     ENSURE CLOSURE
      DUM = 0.
      SUM = 0.
      FAC = 0.
      DO 260 L = 1,NMP
      DUM = DUM -TT(L)
      SUM = SUM-TT(L)*CX(L)
  260 FAC = FAC+TT(L)*SX(L)
      DUM = DUM/FLOAT(NMP)
      DA = 1.-EPSIL-(DX*SIN(DUM)+DY*COS(DUM))/SCALE-FAC/FLOAT(LC)
      DB = (DY*SIN(DUM)-DX*COS(DUM))/SCALE-SUM/FLOAT(LC)
      DO 270 L = 1,NMP
  270 TT(L) = TT(L)+DA*SX(L)-DB*CX(L)
C     FIND THE CONJUGATE FUNCTION DS
      CALL CONJ(NMP,TT,DS,XX,BB,AA)
      DO 290 I = 1,NMP
      SUM = DS(I)
  290 DS(I) = YY(I)*EXP(SUM)
      DS(MC) = DS(1)
      Z(1)=0.
      VAL=.5*PILC
      VAL1=PILC/3.
      Z(2)=VAL*(DS(1)+DS(2))
      NI=NFC+1
      DO 295 J=3,NI,2
      Z(J)=Z(J-2)+VAL1*(DS(J-2)+4.*DS(J-1)+DS(J))
      IF(J.EQ.NI) GO TO 296
  295 Z(J+1)=Z(J)+VAL*(DS(J)+DS(J+1))
  296 CONTINUE
      Z(MC)=0.
      Z(MC-1)=VAL*(DS(MC)+DS(MC-1))
      NII=NFC-2
      DO 299 J=2,NII,2
      MCJ=MC-J
      Z(MCJ)=Z(MCJ+2)+VAL1*(DS(MCJ+2)+4.*DS(MCJ+1)+DS(MCJ))
  299 Z(MCJ-1)=Z(MCJ)+VAL*(DS(MCJ)+DS(MCJ-1))
  300 CONTINUE
      Z1=Z(MC-NII)+VAL1*(DS(MC-NII)+4.*DS(MC-NII-1)+DS(MC-NII-2))
      Z1=Z(NI+1)-Z1
      DO 301 J=3,NI,2
      DS1=Z(NFC+J)-Z(NFC+J-1)
```

```
      Z(NFC+J-1)=Z(NFC+J-2)-Z1
      IF(J.EQ.NI) GO TO 303
      Z1=Z(NFC+J+1)-Z(NFC+J)
  303 CONTINUE
      Z(NFC+J)=Z(NFC+J-1)-DS1
  301 CONTINUE
      SCALE = ARC/Z(MC)
      ERR = 0.
      DO 310 I = 1,NMP
      VAL = ACOS(1.-2.*Z(I)/Z(MC))
      ERR = AMAX1(ERR,ABS(SP(I)-VAL))
  310 SP(I) = VAL
      IF (FSYM.LE.5.) WRITE (N4,490) ERR,DA,DB
      IF (ERR.LT.TOL) GO TO 330
  320 CONTINUE
      WRITE (N4,450)
  330 CALL FOUCF(NMP,TT,CX,BB,AA)
      AA(1) = ARC
      AA(2) = 1.-EPSIL-(DX*SIN(BB(1))+DY*COS(BB(1)))/SCALE
      BB(2) = (-DX*COS(BB(1))+DY*SIN(BB(1)))/SCALE
      IF (FSYM.GT.5.) GO TO 342
      WRITE (N4,460) EPSIL, NMP
      IF ((FSYM.NE.1.).AND.(FSYM.NE.3.)) GO TO 341
      DO 344 L = 1,MM
  344 Z(L) = FLOAT(L-1)*DT
      CALL SPLIF(MC,CIRC,SP,U,V,W,3,0.,3,0.)
      CALL INTPL(MM,Z,DS,CIRC,SP,U,V,W)
      CALL SPLIF (NT,SS,QSR,U,V,W,1,0.,1,0.)
      CALL INTPL(MM,DS,A,SS,QSR,U,V,W)
      DO 4 L = 1,MM
    4 IF (A(L).LE.0.) A(L) = 0.
  341 IF (IZ.NE.120) GO TO 342
      WRITE (N4,540)
      DO 340 L = 1,NFC
  340 WRITE (N4,490) AA(L),BB(L)
  342 CALL MAP
      RETURN
  350 IF (FSYM.LE.5.) GO TO 355
      DXDS1 = (XX(2)-XX(1))/SS(2)
      DXDS2 = (XX(NT)-XX(NT-1))/(SS(NT)-SS(NT-1))
      DYDS1 = (YY(2)-YY(1))/SS(2)
      DYDS2 = (YY(NT)-YY(NT-1))/(SS(NT)-SS(NT-1))
  355 CALL SPLIF(NT,SS,XX,U,SP,W,1,DXDS1,1,DXDS2)
      CALL SPLIF(NT,SS,YY,V,TT,DS,1,DYDS1,1,DYDS2)
      IF (IS.LT.0) GO TO 397
      DC = PI/FLOAT(NMP)
      ERR = SS(NL)
      DUM = DIS(0.)
      FAC = PI/(DIS(PI)-DUM)
      DO 360 L = 1,MC
  360 CIRC(L) = FAC*(DIS(FLOAT(L-1)*DC)-DUM)
      CALL INTPL(NMP,CIRC,SX,SS,XX,U,SP,W)
      CALL INTPL(NMP,CIRC,CX,SS,YY,V,TT,DS)
      SX(MC) = XX(NT)
```

```
      CX(MC) = YY(NT)
      SFAC = 1./(XX(NT)-XX(NL))
      XXNL = XX(NL)
      DO 370 L = 1,MC
      CX(L) = SFAC*CX(L)
      SX(L) = SFAC*(SX(L)-XXNL)
      XX(L) = SX(L)
  370 YY(L) = CX(L)
      WRITE (N4,520) IS
      IF (N2.NE.N4) WRITE (N2,520) IS
      IF(IS.EQ.0) GO TO 395
C     DO  IS  SMOOTHING ITERATIONS
      DO 390 K = 1,IS
      DO 380 L = 2,NMP
      XX(L) = SMOOTH(SX(L-1),SX(L),SX(L+1),SX(L))
  380 YY(L) = SMOOTH(CX(L-1),CX(L),CX(L+1),SX(L))
      DO 390 L = 2,NMP
      SX(L) = XX(L)
  390 CX(L) = YY(L)
  395 NT = MC
      CALL SPLIF(NT,CIRC,XX,U,SP,W,1,0.,1,0.)
      CALL SPLIF(NT,CIRC,YY,V,TT,DS,1,0.,1,0.)
  397 ISYM = 0
      IF (FSYM.GT.5.) GO TO 170
      U(1) = SP(1)
      V(1) = TT(1)
      U(NT) = SP(NT)
      V(NT) = TT(NT)
      GO TO 170
  410 FORMAT (1X16A4,I4)
  420 FORMAT (5F10.7)
  430 FORMAT (35H0AIRFOIL COORDINATES AND CURVATURES/1H0,6X,1HX,14X1HY
     1 ,9X,10HARC LENGTH,7X3HANG,8X5HKAPPA,10X,2HKP,11X,3HKPP//)
  440 FORMAT (1H1,4X,3HERR,14X,2HDA,14X,2HDB//)
  450 FORMAT (32H FOURIER SERIES DID NOT CONVERGE)
  460 FORMAT (34H0MAPPING TO THE INSIDE OF A CIRCLE//3X11HDZ/DSIGMA =
     1 50H -(1/SIGMA**2)*(1-SIGMA)**(1-EPSIL)*(EXP(W(SIGMA))//3X,
     242HW(SIGMA) = SUM((A(N)-I*B(N))*SIGMA**(N-1))//3X,7HEPSIL =
     3 F5.3,20X,I4,25H POINTS AROUND THE CIRCLE )
  470 FORMAT (1H1)
  480 FORMAT (F12.6,2F14.6,F14.3,F14.4,2E14.3)
  490 FORMAT (3E15.6)
C     ****CHANGE (40O1) TO (20A4) ON IBM 360****
  500 FORMAT (40O1)
  510 FORMAT (3X,F4.3,8X,F5.3,8X,F5.3,10X,F4.3,14X,I5)
  520 FORMAT (10H0THERE ARE,I4,26H SMOOTHING ITERATIONS USED /)
  530 FORMAT(4HAIRF,6X,3HOIL,7X,I2,1H-,I1,6X,I1,1H-,2I1)
  540 FORMAT (//7X4HA(N),10X4HB(N)//)
      END
```

```
      SUBROUTINE MAP
C     SUM UP FOURIER SERIES TO OBTAIN MAPPING FUNCTION
      COMPLEX TT,TMP
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
C     ****CHANGE TO 1.E-6 FOR SINGLE PRECISION IBM 360****
      DATA POW,TOL/-12.,10.E-12/
C     NOTE THAT THE SQUARE OF THE MAPPING MODULUS IS BEING COMPUTED
      MX = M/2
C     SET THE SINES AND COSINES
      CO(1) = 1.
      SI(1) = 0.
      DO 5 L = 1,MX
      CO(L+1) = CO(L)*DCN-SI(L)*DSN
      CO(MM-L) = CO(L+1)
      SI(L+1) = CO(L)*DSN+SI(L)*DCN
    5 SI(MM-L) = -SI(L+1)
C     SET MAPPING MODULUS FOR CUSP AT THE TAIL
      DO 10 J = 1,N
      FP (1,J) = 1.+R(J)*(R(J)-2.)
      DO 10 L = 1,MX
   10 FP(L+1,J) = 1.+R(J)*(R(J)-2.*CO(L+1))
      IF (EPSIL.EQ.0.) GO TO 30
C     ADJUST IF THERE IS AN ANGLE AT THE TAIL
      DO 20 J = 1,N
      FP(1,J) = FP(1,J)**(1.-EPSIL)
      DO 20 L = 1,MX
   20 FP(L+1,J) = FP(L+1,J)**(1.-EPSIL)
C     NOW COMPUTE CONTRIBUTION FROM FOURIER SERIES
   30 DO 50 J = 1,N
      NFCX = MINO(NFC,1+INT(POW/ALOG10(R(J)-TOL)))
      RJ = 2.*R(J)
      K = NFCX
      S = AA(K+1)
   35 S = R(J)*S+AA(K)
      K = K-1
      IF (K.GT.1) GO TO 35
      FP(1,J) = FP(1,J)*EXP(S*RJ)
      DO 50 L = 1,MX
      K = NFCX
      LX = K*L
      LT = MOD(LX,M)
      S = AA(K+ )*CO(LT+1)
      Q = BB(K+1)*SI(LT+1)
```

```
   40 LX = LX-L
      LT = MOD(LX,M)
      S = R(J)*S+AA(K)*CO(LT+1)
      Q = R(J)*Q+BB(K)*SI(LT+1)
      K = K-1
      IF (K.GT.1) GO TO 40
      DUM = FP(L+1,J)
      FP(MM-L,J) = EXP(RJ*(S-Q))*DUM
   50 FP(L+1,J) = EXP(RJ*(S+Q))*DUM
      DO 65 L = 1,M
      S = PI-BB(1)
      DO 60 K = 1,NFC
      LT = MOD((L-1)*K,M)
   60 S = S+AA(K+1)*SI(LT+1)-BB(K+1)*CO(LT+1)
      ANG = FLOAT(L-1)*DT
      FP(L,NN) = 1.
   65 FM(L) = S-.5*(ANG+EPSIL*(ANG-PI))
      FM(MM) = FM(1)-(1.+EPSIL)*PI
      DO 70 J = 1,NN
      FP(MM,J) = FP(1,J)
   70 FP(MM+1,J) = FP(2,J)
C     COMPUTE ARC LENGTH AND BODY FROM THE MAPPING BY INTEGRATION
      XMIN = 0.
      YMIN = 0.
      YMAX = 0.
      S = -SQRT(FP(1,1))
      TMP = CMPLX(S*COS(FM(1)),S*SIN(FM(1)))
      DO 80 L = 1,MM
      Q = SQRT(FP(L,1))
      S = S+Q
      ARCL(L) = S
      S = S+Q
      TT = CMPLX(Q*COS(FM(L)),Q*SIN(FM(L)))
      TMP = TMP+TT
      XC(L) = REAL(TMP)
      YC(L) = AIMAG(TMP)
      XMIN = AMIN1(XMIN,REAL(TMP))
      YMIN = AMIN1(YMIN,AIMAG(TMP))
      YMAX = AMAX1(YMAX,AIMAG(TMP))
      TMP = TMP+TT
   80 CONTINUE
      CHD = 1./(.5*XC(MM)-XMIN)
      TC = (YMAX-YMIN)*CHD
      DO 90 L = 1,MM
      ARCL(L) = CHD*ARCL(L)
      XC(L) = CHD*(XC(L)-XMIN)
   90 YC(L) = CHD*YC(L)
      CHD = CHD/(.5*DT)
      IF (NDES.GE.0) RETURN
      IF (ABS(FSYM).GT.5.) GO TO 100
      ANGO= -RAD*BB(1)
      WRITE (N4,120) TC,ANGO
      IF (N2.NE.N4) WRITE (N2,120) TC,ANGO
      IF (MODE.EQ.0) ALP = (1.+BET)*CL/(8.*PI*CHD)-BB(1)
```

```
  100 CALL COSI
      RETURN
  120 FORMAT (32HOTHE THICKNESS TO CHORD RATIO IS ,F6.4//10H THE ANGLE
     1 17H OF ZERO LIFT IS ,F6.3,8H DEGREES)
      END




      SUBROUTINE SPLIF (N,S,F,FP,FPP,FPPP,KM,VM,KN,VN)
C     SPLINE FIT - SUBROUTINE CONTRIBUTED BY ANTHONY JAMESON
C     GIVEN S AND F AT N CORRESPONDING POINTS,COMPUTE A CUBIC SPLINE
C     THROUGH THESE POINTS SATISFYING AN END CONDITION IMPOSED ON
C     EITHER END.  FP,FPP,FPPP WILL BE THE FIRST,SECOND AND THIRD
C     DERIVATIVE RESPECTIVELY AT EACH POINT ON THE SPLINE
C     KM IS THE DERIVATIVE IMPOSED AT THE START OF THE SPLINE
C     VM WILL BE THE VALUE OF THE DERIVATIVE THERE
C     KN IS THE DERIVATIVE IMPOSED AT THE END OF THE SPLINE
C     VN WILL BE THE VALUE OF THE DERIVATIVE THERE
C     KM,KN CAN TAKE VALUES 1,2, OR 3
C     S MUST BE MONOTONIC
      DIMENSION S(1), F(1), FP(1), FPP(1), FPPP(1)
      K = 1
      M = 1
      I = M
      J = M+K
      DS = S(J)-S(I)
      D = DS
      IF (DS.EQ.0.) CALL ABORT
      DF = (F(J)-F(I))/DS
      IF (IABS(KM)-2) 10,20,30
   10 U = .5
      V = 3.*(DF-VM)/DS
      GO TO 50
   20 U = 0.
      V = VM
      GO TO 50
   30 U = -1.
      V = -DS*VM
      GO TO 50
   40 I = J
      J = J+K
      DS = S(J)-S(I)
      IF (D*DS.LE.0.) CALL ABORT
      DF = (F(J)-F(I))/DS
      B = 1./(DS+DS+U)
      U = B*DS
      V = B*(6.*DF-V)
   50 FP(I) = U
      FPP(I) = V
      U = (2.-U)*DS
      V = 6.*DF+DS*V
      IF (J.NE.N) GO TO 40
      IF (KN-2) 60,70,80
```

```
   60 V = (6.*V .-V)/U
      GO TO 90
   70 V = VN
      GO TO 90
   80 V = (DS*VN+FPP(I))/(1.+FP(I))
   90 B = V
      D = DS
  100 DS = S(J)-S(I)
      U = FPP(I)-FP(I)*V
      FPPP(I) = (V-U)/DS
      FPP(I) = U
      FP(I) = (F(J)-F(I))/DS-DS*(V+U+U)/6.
      V = U
      J = I
      I = I-K
      IF (J.NE.M) GO TO 100
      FPPP(N) = FPPP(N-1)
      FPP(N) = B
      FP(N) = DF+D*(FPP(N-1)+B+B)/6.
      IF (KM.GT.0) RETURN
C     IF KM IS NEGATIVE COMPUTE THE INTEGRAL IN FPPP
      FPPP(J) = 0.
      V = FPP(J)
  105 I = J
      J = J+K
      DS = S(J)-S(I)
      U = FPP(J)
      FPPP(J) = FPPP(I)+.5*DS*(F(I)+F(J)-DS*DS*(U+V)/12.)
      V = U
      IF (J.NE.N) GO TO 105
      RETURN
      END
```

```
      SUBROUTINE INTPL (NX,SI,FI,S,F,FP,FPP,FPPP,
C     GIVEN S,F(S) AND THE FIRST THREE DERIVATIVES AT A SET OF POINTS
C     FIND FI(SI) AT THE NX VALUES OF SI BY EVALUATING THE TAYLOR SERIES
C     OBTAINED BY USING THE FIRST THREE DERIVATIVES
      DIMENSION SI(1), FI(1), S(1), F(1), FP(1), FPP(1), FPPP(1)
      DATA PT/.33333333333333/
      J = 0
      DO 30 I = 1,NX
      VAL = 0.
      SS = SI(I)
   10 J = J+1
      TT = S(J)-SS
      IF (TT) 10,30,20
   20 J = MAX0(1,J-1)
      SS = SS-S(J)
      VAL = SS*(FP(J)+.5*SS*(FPP(J)+SS*PT*FPPP(J)))
   30 FI(I) = F(J)+VAL
      RETURN
      END
```

```
      SUBROUTINE CONJ (N,F,G,X,CN,SN)
C     CONJUGATION BY FAST FOURIER TRANSFORM
C     GIVEN THE REAL PART F OF AN ANALYTIC FUNCTION ON THE UNIT CIRCLE
C     THE IMAGINARY PART G IS CONSTRUCTED
      COMPLEX F,G,EIV,EIT
      DIMENSION   F(1),G(1),X(1),     CN(1),SN(1)
      DATA PI/3.14159265358979/
      L          = N/2
      DX = 1./FLOAT(L)
      EIV = CMPLX(COS(PI*DX),SIN(PI*DX))
      DO 2 I = 1,L
    2 G(I)       = F(I)
      CALL FFORM(L,G,X,CN,SN)
      G(1) = 0.
      I = 1
      DO 10 J = 1,L,2
      EIT = CMPLX(SN(I)*DX,CN(I)*DX)
      I = I+1
      G(J) = G(J)*EIT
   10 G(J+1) = G(J+1)*EIT*EIV
      DO 22 I=1,L
   22 SN(I)      = -SN(I)
      CALL FFORM(L,G,X,CN,SN)
      DO 32 I=1,L
   32 SN(I)      = -SN(I)
      EIV  = CMPLX(AIMAG(G(L)),REAL(G(1)))
      I = L
   40 G(I) = CMPLX(AIMAG(G(I-1)),REAL(G(I)))
      I = I-1
      IF (I.GT.1) GO TO 40
      G(1) = EIV
      RETURN
      END



      SUBROUTINE FOUCF(N,G,X,A,B)
C     FOURIER COEFFICIENTS BY FAST FOURIER TRANSFORM
      COMPLEX G,EIV,QP,X,GK
      DIMENSION       G(1),X(1),     A(1),B(1)
      DATA PI/3.14159265358979/
      L          = N/2
      V          = PI/L
      EIV = CMPLX(COS(V),SIN(V))
      ENI = 1./FLOAT(N)
      CALL FFORM(L,G,X,A,B)
      GK = 0.
      I = 1
      DO 5 J = 1,L,2
      X(J) = CMPLX(B(I),A(I))
      X(J+1) = X(J)*EIV
    5 I = I+1
      K = L
```

```
      DO 10 J = 1,L
      QP = GK-CONJG(G(J))
      GK = GK+CONJG(G(J))-QP*X(J)
      A(J) = -REAL(GK)*ENI
      B(J) = AIMAG(GK)*ENI
      GK = G(K)
   10 K = K-1
      A(L+1) = -B(1)
      B(1)      = 0.
      B(L+1)    = 0.
      RETURN
      END
```

```
      SUBROUTINE FFORM(N,F,X,CN,SN)
C     FAST FOURIER TRANSFORM
C     INPUT ARRAY F WITH REAL AND IMAGINARY PARTS IN ALTERNATE CELLS
C     REPLACED BY ITS FOURIER TRANSFORM
      COMPLEX F(1),X(1),W
      DIMENSION CN(1),SN(1)
      IF (N.LT.2) RETURN
      NS = 1
      NR = 2
      NQ = N
C     SET THE SINES AND COSINES
      PI=3.14159265356979
      DT = (PI+PI)/FLOAT(N)
      IF((SN(1).EQ.0.).AND.(SN(2).EQ.SIN(DT))) GO TO 11
      ANG = 0.
      DO 5 J = 1,N
      CN(J) = COS(ANG)
      SN(J)=-SIN(ANG)
    5 ANG = ANG+DT
   11 DO 10 K = NR,N
      IF (MOD(NQ,K).EQ.0) GO TO 21
   10 CONTINUE
   21 ND=NQ/K
      NS = NS*K
      NR = K
      IQ = 0
      ID = 0
      DO 22 I = 1,NS
      DO 24 J = 1,ND
      L = IQ+J
      LP=L+ND
      M=ID
      W =F(L)+F(LP)*CMPLX(CN(M+1),SN(M+1))
      IF(NR.EQ.2) GO TO 24
      L=LP
      DO 26 K=3,NR
      L = L+ND
      M = M+ID
```

```
      IF (M.GE.N) M = M-N
   26 W = W+F(L)*CMPLX(CN(M+1),SN(M+1))
   24 X(ID+J) = W
      ID = ID+ND
      IQ=IQ+NQ
      IF(IQ.GE.N) IQ=IQ-N
   22 CONTINUE
      NQ = ND
      IF (ND.GT.1) GO TO 61
      DO 32 K = 1,N
   32 F(K) = X(K)
      RETURN
   61 DO 60 K = NR,N
      IF (MOD(NQ,K).EQ.0) GO TO 71
   60 CONTINUE
   71 ND=NQ/K
      NS = NS*K
      NR = K
      IQ = 0
      ID = 0
      DO 72 I = 1,NS
      DO 74 J = 1,ND
      L = IQ+J
      LP=L+ND
      M=ID
      W=X(L)+X(LP)*CMPLX(CN(M+1),SN(M+1))
      IF(NR.EQ.2) GO TO 74
      L=LP
      DO 76 K=3,NR
      L = L+ND
      M = M+ID
      IF (M.GE.N) M = M-N
   76 W = W+X(L)*CMPLX(CN(M+1),SN(M+1))
   74 F(ID+J) = W
      ID = ID+ND
      IQ=IQ+NQ
      IF(IQ.GE.N) IQ=IQ-N
   72 CONTINUE
      NQ = ND
      IF (ND.GT.1) GO TO 11
      RETURN
      END



      FUNCTION INDEXR(X,ARRAY,N)
      DIMENSION ARRAY(1)
      S = ABS(X-ARRAY(N))
      DO 10 L = 1,N
      IF (ABS(X-ARRAY(L)).GT.S) GO TO 10
      INDEXR = L
      S = ABS(X-ARRAY(L))
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GTURB(DELMAX,DELBP,CPO,BCP,SL,RDEL,RBCP)
      COMMON/FL/FLUXT4,CD4,CDW,INDCD
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CU(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DP,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,I2,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,KEM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      REAL MACH,MACHN,NEW,MACHS
      DIMENSION HP(162),SEPP(162),CPP(162),THETAP(162),DELP(162)
     1  ,DELX(1),TD(1)
      DIMENSION H(1),THETA(1),DELS(1),XX(1),YY(1),MACHN(1)
     1 ,SEPR(1),CPX(1),DSDT(1),S(1),MACHS(1),ANGNEW(1)
      EQUIVALENCE (MACHN(1),A(1))    ,(H(1),FP(1,6)),(THETA(1),FP(1,8))
     1            ,(XX(1),FP(1,3 )),(YY(1),FP(1,5)),(DELS(1),FP(1,10))
     2,(ANGNEW(1),FP(1,24)),(SEPR(1),FP(1,14)),(CPX(1),PHIR(1))
     3 ,(S(1),FP(1,16)),(MACHS(1),FP(1,26)),(DSDT(1),FP(1,30))
     4 ,(DELX(1),FP(1,12)),(TD(1),FP(1,20))
      CP(Q) = C5*((C4/(1.+C2*Q*Q))**C7-1.)
      QSX(Q) = (C4-(1.+Q/C5)**(1./C7))/C6
      MACH(Q) = SQRT (Q/(C1-C2*Q))
      DATA ISW/0/,CDF/0./,XPLT/.5/,XFAC/100./
      IF (NDES.GE.1) GO TO 5
      DO 10 J = 1,NN
      PHI(MM,J) = PHI(1,J)+DPHI
   10 PHI(MM+1,J) = PHI(2,J)+DPHI
    5 IF (ISW.EQ.0 .AND.CSTAR.EQ.100.) CALL GOPLOT(NRN)
C     COMPUTE AND STORE CP CRITICAL
      CPX(MM+1) = CP(1.)
C     ISX SET TO 1 FOR FSYM=1. AND FSYM=3 IF FLOW HAS NOT BEEN COMPUTED
      ISX = (NCY+1)*(ITYP-3)*ABS(FSYM+10.)+.2
      IF (ISX.NE.1) GO TO 30
      M4 = N3
      FSYM = 0.
      ALP = 0.
      XSEP = AMAX1(0.,XSEP-1.)
      QS = A(MM)
      DO 20 L = 1,MM
      XOLD(L) = XC(L)
      YOLD(L) = YC(L)
      MACHN(L) = MACH(A(L))
   20 CPX(L) = CP(MACHN(L))
      IF ((ABS(YC(MM)-YC(1)).LE.1.E-5).AND.(IABS(NRN).GT.999)) GO TO 50
      GO TO 110
   30 DO 40 L = 2,M
      U = (PHI(L+1,1)-PHI(L-1,1))*DELTH-SI(L)
      QS = (U*U)/FP(L,1)
      MACHN(L) = MACH(QS)
```

```
   40 CPX(L) = CP(MACHN(L))
      MACHN(MM) = .5*(MACHN(2)+MACHN(M))
      MACHN(1) = MACHN(MM)
      CPX(1) = CP(MACHN(1))
      CPX(MM) = CPX(1)
      QS=QSX(CPX(MM))
      IF((INDCD.EQ.1).AND.(FSYM.EQ.7)) GO TO 50
      IF (FSYM.EQ.6.) GO TO 60
      IF ((FSYM.LE.5.).OR.(ITYP.LE.2)) GO TO 50
C     ADVANCE PLOTTER PAPER TO THE NEXT BLANK PAGE
      IF(XPLT.GT..5) CALL PLOT(12.0*FLOAT(INT((20.2+XPLT)/12.)),0.,-3)
      XPLT = .5
   50 CALL GETCP(CDF)
      IF(INDCD.EQ.1) ISW=1
      IF(INDCD.EQ.1) RETURN
      CALL GOPRIN (HP,THETAP,SEPP,CPP,DELP,XTRANS)
      IF (ISX.EQ.1) CALL EXIT
      ISW = 1
      RETURN
   60 DO 70 L = 1,MM
   70 CPP(L) = CPX(L)
      IF((ISW.EQ.0).OR.(FSYM.NE.6.)) GO TO 90
C     FIND THE BASE PRESSURE
      DELBP = 10.
      CPO = CP(MACHN(IXX-1))
      DO 80 L = IXX,M
      CPN = CP(MACHN(L))
      DELBP = AMIN1(DELBP,CPN-CPO)
   80 CPO = CPN
      BCP = BCP+RBCP*DELBP
   90 ISW = 1
      PCH = ABS(PCH)
      IF (LSEP.GE.MM) GO TO 110
C     MODIFY THE MACH DISTRIBUTION
      CPO = CP(MACHN(LSEP))
      SEPX = XC(LSEP)
      SL = (BCP-CPO)/(XC(MM)-SEPX)
      DO 100 L = LSEP,MM
      CPP(L) = CPO+SL*(XC(L)-SEPX)
  100 MACHN(L) = MACH(QSX(CPP(L)))
  110 KQMIN = 1
      KQMAX = 1
      QMIN = MACHN(1)
      QMAX = QMIN
      DARC = TP/FLOAT(NPTS-1)
      DO 115 L = 1,NPTS
  115 H(L) = FLOAT(L-1)*DARC
      H(NPTS) = TP
      DO 116 L = 1,M
  116 YY(L) = FLOAT(L-1)*DT
      YY(MM) = TP
      CALL SPLIF (MM,YY,ARCL,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL (NPTS,H,S,YY,ARCL,DSDT,CO,TD)
      S(NPTS) = ARCL(MM)
```

```
      CALL SPLIF (MM,ARCL,MACHN,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL(NPTS,S    ,MACHS,ARCL,MACHN,DSDT,CO,TD)
      CALL SPLIF (MM,ARCL,XC,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL (NPTS,S,XX,ARCL,XC,DSDT,CO,TD)
      DO 120 L = 1,NPTS
      IF (MACHS(L).GT.QMAX) KQMAX = L
      IF (MACHS(L).LT.QMIN) KQMIN = L
      QMIN = AMIN1(MACHS(L),QMIN)
      QMAX = AMAX1(MACHS(L),QMAX)
      SEPR(L) = 0.
      H(L) = 0.
      DELS(L) = 0.
  120 THETA(L) = 0.
      IF (PCH.LT.0.) GO TO 140
      KQMAX = KQMIN+INDEXR(PCH,XX(KQMIN+1),NPTS-KQMIN)
      IF (KQMAX.GE.NPTS) CALL ABORT
  140 CALL NASHMC (KQMAX,NPTS)
      XTRANS = PCH
      IF (PCH.LT.0) XTRANS = XX(KQMAX)
      KQBOT = INDEXR(XTRANS,XX,KQMIN)
      IF (KQBOT.LE.1) CALL ABORT
      CALL NASHMC (KQBOT,1)
      FAC=S(4)/(S(4)-S(2))
      THETA(1)=FAC*THETA(2)+(1.-FAC)*THETA(4)
      H(1)=FAC*H(2)+(1.-FAC)*H(4)
      DELS(1)=H(1)*THETA(1)
      IF(XSEP.GE.0.) GO TO 141
      FAC=(S(NPTS-3)-S(NPTS))/(S(NPTS-3)-S(NPTS-1))
      THETA(NPTS)=FAC*THETA(NPTS-1)+(1.-FAC)*THETA(NPTS-3)
      H(NPTS)=FAC*H(NPTS-1)+(1.-FAC)*H(NPTS-3)
      DELS(NPTS)=H(NPTS)*THETA(NPTS)
  141 CONTINUE
C     COMPUTE THE SKIN FRICTION DRAG
      Q = SQRT(QS)
      RT = (C1-C2*QS)/(C1-C2)
      HBT = (H(NPTS)+1.)*(1.-C2*QS/C1)-1.
      HBB = (H(1)+1.)*(1.-C2*QS/C1)-1.
      CDF = 2.*THETA(NPTS)*Q**(.5*(  HBT  +5.))*RT**3
      CDF = CDF+2.*THETA(1)*Q**(.5*( HBB+5.))*RT**3
      IF (ISX.EQ.1) GO TO 200
C     MAKE DISPLACEMENT MONOTONE INCREASING ON THE UPPER SURFACE
      DO 170 L = KQMAX,NPTS
      IF (DELS(L+1).LT.DELS(L)) DELS(L+1) = DELS(L)
  170 CONTINUE
C     LOWER SURFACE - FIND WHERE DELS STARTS DECREASING
C     TREAT THE LOWER SURFACE LIKE THE UPPER SURFACE IF XSEP.LT.0
      XPC = .60
      IF (XSEP.LT.0.) XPC = 2.
      J = KQBOT
  180 J = J-1
      IF (DELS(J-1).LT.DELS(J)) GO TO 185
      IF (J.GE.2) GO TO 180
      GO TO 200
  185 IF (XX(J).GT.XPC) GO TO 190
```

```
      DELS(J-1) = DELS(J)
      GO TO 180
C     DISPLACEMENT MUST STAY MONOTONE DECREASING
  190 J = J-1
      IF (DELS(J-1).GT.DELS(J)) DELS(J-1) = DELS(J)
      IF (J.GT.2) GO TO 190
C     SMOOTH DELS  IS  TIMES
  200 IF (IS.LE.0) GO TO 220
      DO 210  I = 1,IS
      OLD = DELS(1)
      DO 210 L = 3,NPTS
      NEW = DELS(L-1)
      DELS(L-1) = .25*(OLD+NEW+NEW+DELS(L))
  210 OLD = NEW
  220 XPLT = XPLT+.5
      FAC=(S(NPTS-1)-S(NPTS))/(S(NPTS-1)-S(NPTS-2))
      DELS(NPTS)=FAC*DELS(NPTS-2)+(1.-FAC)*DELS(NPTS-1)
      IF(XSEP.GE.0.) GO TO 221
      FAC=(S(2)-S(1))/(S(2)-S(3))
      DELS(1)=FAC*DELS(3) (1.-FAC)*DELS(2)
  221 CONTINUE
      IF (ISX.EQ.1) GO TO 260
      YFAC = 10./S(NPTS)
      DH = (H(KQMAX+1)-H(KQBOT-1))/ FLOAT(2+KQMAX-KQMIN)
      FAC = ARCOLD(NT)/S(NPTS)
      IF(XPLT.LT.1.2) CALL SYMBOL(.35,8.74,.14,55HDISPLACEMENT THICKNESS
     1 AT EACH BOUNDARY LAYER ITERATION,270.,55)
      CALL PLOT (XPLT+XFAC*DELS(1),10.5,3)
      DO 230 L = 1,NPTS
      CALL PLOT(XPLT+XFAC*DELS(L),10.5-YFAC*S(L),2)
      IF ((L.GE.KQBOT).AND.(L.LE.KQMAX)) H(L) = H(L-1)+DH
  230 YY(L) = S(L)*FAC
      YY(NPTS) = ARCOLD(NT)
C     DELX WILL BE BOUNDRY LAYER DISPLACEMENT AT NT POINTS
      CALL SPLIF(NPTS,YY,DELS,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL(NT,ARCOLD,DELX,YY,DELS,DSDT,CO,TD)
C     THE FOLLOWING ARE BEING COMPUTED FOR FUTURE PRINT OUT
      CALL SPLIF(NPTS,S,DELS,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL(MM,ARCL,DELP,S,DELS,DSDT,CO,TD)
      CALL SPLIF (NPTS,S,H,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL(MM,ARCL,HP,S,H,DSDT,CO,TD)
      CALL SPLIF(NPTS,S,THETA,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL (MM,ARCL,THETAP,S,THETA,DSDT,CO,TD)
      CALL SPLIF(NPTS,S, SEPR,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL(MM,ARCL,SEPP,S,SEPR,DSDT,CO,TD)
C     GET THE SLOPES FOR THE OUTER AIRFOIL AT CORRESPONDING POINTS
      DO 240 L = 1,MM
      DDEL = RDEL*(DELP(L)-DSUM(L))
      DELP(L) = DDEL
      DSUM(L) = DSUM(L)+DDEL
  240 S(L) = FAC*ARCL(L)
      S(MM) = ARCOLD(NT)
      CALL SPLIF(MM,S,FM,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL(NT,ARCOLD,ANGNEW,S,FM,DSDT,CO,TD)
```

```
      DELMAX = 0.
      DO 250 L = 1,NT
      DDEL = DELX(L)-DELOLD(L)
      DELMAX = AMAX1(DELMAX,ABS(DDEL))
      DY = DELOLD(L)+RDEL*DDEL
      ANG = .5*(ANGOLD(L)+ANGNEW(L))
      XX(L)=XOLD(L)
      YY(L)=YOLD(L)+DY/COS(ANG)
  250 DELOLD(L) = DY
      ISS = IS
      IS = -1
      IF (ITYP.EQ.99) CALL GOPRIN (HP,THETAP,SEPP,CPP,DELP,XTRANS)
      CALL AIRFOL
      IS = ISS
      FSYM = 7.
      RETURN
  260 DO 270 L = 1,MM
      ARCOLD(L) = ARCL(L)
      CPP(L) = CPX(L)
  270 ANGOLD(L) = FM(L)
      CALL SPLIF(NPTS,S,DELS,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL(MM,ARCL,DSUM,S,DELS,DSDT,CO,TD)
      CALL SPLIF(NPTS,S,SEPR,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL (MM,ARCL,SEPP,S,SEPP,DSDT,CO,TD)
      CALL SPLIF (NPTS,S,THETA,DSDT,CO,TD,3,0.,3,0.)
      CALL INTPL (MM,ARCL,THETAP,S,THETA,DSDT,CO,TD)
      CALL GOPRIN (HP,THETAP,SEPP,CPP,DELP,XTRANS)
      NT = MM
      CALL GETCP(CDF)
      IF (JK.LE.-1 ) CALL PLOT (0.,0.,999)
      CALL EXIT
      END


      SUBROUTINE GOPRIN(H,THETA,SEP,CPP,DEL,XTR)
      REAL MACHN
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),R1(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,PN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,RFM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      DIMENSION DSDT(1),FPP(1),FPPP(1),H(1),SEP(1),THETA(1),CPP(1)
     1 ,MACHN(1),CP(1),DEL(1),BL(4)
      EQUIVALENCE (FPP(1),CO(1)),(FPPP(1),SI(1)),(DSDT(1),FP(1,31))
      EQUIVALENCE (MACHN(1),A(1)),(CP(1),PHIR(1))
```

```
      DATA ION,IOFF,Z,SEPMAX/1,0,0.,.004/
      SN = -2./ARCL(MM)
      QMIN=MACHN(1)
      DO 10 L = 1,M
      QMIN=AMIN1(MACHN(L),QMIN)
   10 ARCL(L) = ACOS(1.+SN*ARCL(L))
      ARCL(MM) = PI
      CALL SPLIF(MM,ARCL,FM,DSDT,FPP,FPPP,1,0.,1,0.)
      DSDT(1) = FPP(1)*1.E-5
      DSDT(MM) = -FPP(MM)*1.E-5
      DO 20 L = 1,MM
      FPP(L) = RAD*FM(L)-180.
   20 FPPP(L)= SN*DSDT(L)/AMAX1(1.E-5,SIN(ARCL(L)))
      IF (FSYM.GT.5.) GO TO 120
      IF (FSYM.EQ.0.) GO TO 60
      WRITE (N4,310)
   25 IF (FSYM.EQ.0) WRITE (N4,320) TTLE
      IF (XP.EQ.0.) WRITE(N4,360) IOFF
      IF (XP.NE.0.) WRITE(N4,660) IOFF
      IF (XP.EQ.0.) GO TO 600
      REWIND M4
      READ (M4,555)
      READ (M4,555)
  555 FORMAT (1H1)
  600 DO 30 L = 1,MM
      IF (XP.EQ.0.) GO TO 610
      IF (MOD(L+1,55).EQ.0) WRITE (N4,660) ION
      READ(M4,556) DUM,DUM
  556 FORMAT(2F10.4)
      WRITE(N4,670) L,XC(L),YC(L),FPP(L),FPPP(L),MACHN(L),CP(L),DUM
      GO TO 30
  610 IF (MOD(L+1,55).EQ.0) WRITE (N4,360) ION
      WRITE(N4,260) L,XC(L),YC(L),FPP(L),FPPP(L),MACHN(L),CP(L)
   30 CONTINUE
  670 FORMAT (I14,2F9.5,2F8.2,3F9.4)
C     RESTORE QUANTITIES TO VALUES THEY HAD UPON ENTERING THIS ROUTINE
   40 DO 50 L = 1,MM
      ARCL(L) = (COS(ARCL(L))-1.)/SN
   50 FP(L,NN) = 1.
      CALL COSI
      RETURN
   60 RNX = .1*AINT(RN*1.E-5)
      IF ((ABS(YC(MM)-YC(1)).LE.1.E-5).AND.(IABS(NRN).GT.999)) GO TO 25
      WRITE (N4,390) TTLE,RNX
      WRITE (N4,330) IOFF
      IF ( JK.GE.0 ) GO TO 80
      CALL PLOT (2.,0.,-3)
      ENCODE (30,370,TTLE) EM,CL,TC
      CALL SYMBOL (1.2,.7,.14,TTLE,0.,30)
      ENCODE (20,380,TTLE) RNX
      CALL SYMBOL (1.5,1.0,.14,TTLE,0.,20)
      CALL PLOT(PLTSZ*XC(1),5.+PLTSZ*YC(1),3)
      DO  70 L = 2,MM
   70 CALL PLOT(PLTSZ*XC(L),5.+PLTSZ*YC(L),2)
```

```
      IPEN = 3
   80 DO 100 L = 1,MM
      XS = XOLD(L)+DSUM(L)*SIN(ANGOLD(L))
      YS = YOLD(L)-DSUM(L)*COS(ANGOLD(L))
      XC(L) = XS
      YC(L) = YS
      IF (JK.LE.-1)  CALL PLOT(PLTSZ*XS,5.+PLTSZ*YS,IPEN)
      IPEN = 2
      IF (MOD(L+3,55).EQ.0) WRITE (N4,330) ION
      IF (XOLD(L).GT.XTR) GO TO 90
      TRANS = 1H
      IF (MACHN(L).EQ.QMIN) TRANS = 10HSTAGNATION
      IF ((XOLD(L+1).GT.XTR).OR.(XOLD(L-1).GT.XTR)) GO TO 85
      IF ((XOLD(L+2).GT.XTR).OR.(XOLD(L-2).GT.XTR))TRANS= 10HTRANSITION
   85 WRITE (N4,340) XOLD(L),YOLD(L),FPP(L),FPPP(L),CPP(L),TRANS,XS,YS
      GO TO 100
   90 WRITE (N4,350) XOLD(L),YOLD(L),FPP(L),FPPP(L),CPP(L),THETA(L)
     1 ,SEP(L),XS,YS
  100 CONTINUE
      IF (XP.EQ.0.) NRN = -IABS(NRN)
      XP = -ABS(XP)
      RETURN
  120 WRITE (N4,310)
      WRITE (N4,300) IOFF
      I = 1
      YSEP = ABS(XSEP)
      IF (XSEP.GT.0.) YSEP = 2.
      DO 150 L = 1,MM
      IF (MOD(L,55).EQ.0) WRITE (N4,300) ION
      IF (XC(L).GT.XTR) GO TO 130
      TRANS = 1H
      IF (MACHN(L).EQ.QMIN) TRANS = 10HSTAGNATION
      IF ((XC(L+1).GT.XTR).OR.(XC(L-1).GT.XTR)) GO TO 125
      I = -1
      YSEP = ABS(XSEP)
      IF ((XC(L+2).GT.XTR).OR.(XC(L-2).GT.XTR)) TRANS = 10HTRANSITION
  125 WRITE (N4,290)          L,XC(L),YC(L),FPP(L),FPPP(L),MACHN(L),
     1 CP(L),CPP(L),Z,Z,TRANS,L
      GO TO 150
  130 BL(1) = 1H
      BL(2) = 1H
      BL(3) = 1H
      BL(4)= 1H
      IF (L.EQ.LSEP)   BL(1) = 2HLS
      IF((SEP(L).GT.SEPMAX).AND.(SEP(L+I).LT.SEPMAX)) BL(2)= 2HCS
      IF (L.EQ.IXX) BL(3)= 2HLM
      IF((XC(L).GE.YSEP).AND.(XC(L+I).LT.YSEP)) BL(4) = 2HLP
      WRITE (N4,280) BL    ,L,XC(L),YC(L),FPP(L),FPPP(L),MACHN(L),
     1 CP(L),CPP(L),THETA(L),DSUM(L),SEP(L),H(L),DEL(L),L
  150 CONTINUE
      GO TO 40
  260 FORMAT(I14,2F9.5,2F8.2,2F9.4)
  280 FORMAT(3X,  4A2,I5,2F9.5,F9.2,F8.2,F8.4,2F9.4,F9.5,F9.5,F9.5,F7.2,
     1E9.2,I5)
```

```
290 FORMAT (I16,2F9.5,F9.2,F8.2,F8.4,2F9.4,2F9.5,8X,A10,7X,I5)
300 FORMAT(I1,14X1HL5X2HXS,7X,2HYS,7X,3HANG,4X,5HKAPPA,4X,4HMACHcX24C
   1 ,6X3HCP1,4X5HTHETA,5X4HDELS,6X3HSEP,6X1HH,5X2HDD,6X1HL/)
310 FORMAT(1H1,15X,40HLOWER SURFACE TAIL TO UPPER SURFACE TAIL )
320 FORMAT(1H1/  17X26HLISTING OF COORDINATES FOR,2X,4A4)
330 FORMAT(I1 /11X1HX,8X,1HY,6X,3HANG,4X,5HKAPPA,6X,2HCP,5X,
   1 5HTHETA,5X,3HSEP,6X,2HXS,7X,2HYS/)
340 FORMAT (F14.5,F9.5,F8.2,F8.2,F9.4,4X,A10,4X,2F9.5)
350 FORMAT (F14.5,F9.5,F8.2,F8.2,F9.4,4F9.5)
360 FORMAT (I1/12X1HL,6X,1HX,8X,1HY,6X,3HANG,4X5HKAPPA4X4HMACHcX2HCP/.
370 FORMAT ( 2HM=,F4.3,4X,3HCL=,F5.3,4X,4HT/C=,F4.3)
380 FORMAT (4H RN=,F4.1,9H  MILLION )
390 FORMAT(1H1/   9X26HLISTING OF COORDINATES FOR,2X,4A4 ,4X,3HRN=,
   1 F4.1,8H MILLION )
660 FORMAT (I1/12X1HL,6X,1HX,8X,1HY,6X,3HANG,4X5HKAPPA4X4HMACHcX2HCP,
   1 8X,4HDATA/)
    END



      SUBROUTINE NASHMC (K1,K2)
C     COMPUTE THE BOUNDRY LAYER FROM POINT K1 TO K2
C     K3 WILL BE THE SEPARATION POINT
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CD(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELCLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM, LP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DF,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,CCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,N1,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NOPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,CPU
      DIMENSION MACHS(1),H(1),THETA(1),SEPR(1),S(1),DELS(1),XX(1)
      EQUIVALENCE(MACHS(1),FP(1,28)),(H(1),FP(1,6)),(THETA(1),FP(1,8))
      EQUIVALENCE (SEPR(1),FP(1,14)),(DELS(1),FP(1,10)),(S(1),FP(1,16))
      EQUIVALENCE (XX(1),FP(1,3))
      REAL MH,MHSQ,NU,MACHS
      DATA TR,RTHO,TE1,TE2,SEPMAX,PIMIN,PIMAX /.3424,320.,5.E-3,5.E-5,
     1 .004,-1.5,1.E4/
      GAM1 = .5/C2
      CSIINF = C4
      INC = ISIGN(1,K2-K1)
      YSEP = ABS(XSEP)
      IF ((XSEP.GT.0.).AND.(INC.LT.0)) YSEP = 1.
      SEPMAX = SEPM
      GE = 6.5
      L = K1
      DS = ABS(S(L)-S(L-INC))
   10 LP = L+INC
      MH = .5*(MACHS(L)+MACHS(LP))
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      MHSQ = MH*MH
      CSIH = 1.+C2*MHSQ
      DSOLD = DS
      DS = ABS(S(LP)-S(L))
      DQDS = (MACHS(LP)-MACHS(L))/(DS*MH*CSIH)
      T = CSIINF/CSIH
      RHOH = T**GAMI
      NU = T*(1.+TR)/(RHOH*(T+TR))
      RTH = RN*MH/(EM*NU)
      IF (L.NE.K1) GO TO 30
      THETAH= RTHO/RTH
      THT = THETAH
   30 FC = 1.0+.066*MHSQ-.008*MH*MHSQ
      FR = 1.-.134*MHSQ+.027*MHSQ*MH
C     DO AT MOST 200 ITERATIONS
      DO 140 J = 1,499
      RTAU= 1./(FC*(2.4711*ALOG(FR*RTH*THETAH)+4.75)+1.5*GE+1724./
     1  (GE*GE+200.)-16.87)
      TAU = RTAU*RTAU
      HB = 1./(1.-GE*RTAU)
      HH   = (HB+1.)*(1.+.178*MHSQ)-1.
      SEP = -THETAH*DQDS
      IF (SEP.LT.SEPMAX) GO TO 50
      IF (XX(L).LT.YSEP) SEP = SEPMAX
   50 PIE = HH*SEP/TAU
      PIE = AMAX1(PIMIN,AMIN1(PIMAX,PIE))
      G = 6.1*SQRT(PIE+1.81)-1.7
      T2 = ABS(G-GE)/GE
      GE = G
      DT2 = DT1
      DT1 = (HH+2.-MHSQ)*SEP+TAU
      IF (J.EQ.1) GO TO 110
      T1 = ABS((DT1-DT2)/DT1)
      IF ((T1.LT.TE2).AND.(T2.LT.TE1)) GO TO 130
  110 THETAH = THT+.5*DT1*DS
  140 CONTINUE
  130 THETA(LP) = THT+DT1*DS
      SEP = -THETAH*DQDS
      THETAH = THETA(LP)
      THT = THETA(LP)
      SEPR(L) = (SEPR(L)*DS+SEP*DSOLD)/(DS+DSOLD)
      SEPR(LP) = SEP
      H(L) = (H(L)*DS+HH*DSOLD)/(DS+DSOLD)
      H(LP) = HH
      DELS(L) = H(L)*THETA(L)
      L = LP
      IF (L.NE.K2) GO TO 10
      H(K2)=H(K2-INC)+(DS/DSOLD)*(H(K2-INC)-H(K2-INC-INC))
      SEPR(K2) = 2.*SEPR(K2)-SEPR(K2-INC)
      DELS(K2) = H(K2)*THETA(K2)
      H(K1) = 0.
      SEPR(K1) = 0.
      CALL NASHLS(K2)
      DELS(K2-INC) = H(K2-INC)*THETA(K2-INC)
```

```
      DELS(K2) = H(K2)*THETA(K2)
      RETURN
      END



      SUBROUTINE TRIDI(A,B,C,RHS,OUT,N,IDIM)
      COMPLEX RHS,OUT
      DIMENSION A(1),B(1),C(1)
      DIMENSION GA(35),RHS(IDIM),OUT(35)
      REC=1./B(1)
      GA(1)=REC*C(1)
      OUT(1)=REC*RHS(1)
      DO 10 J=2,N
      REC=1./(B(J)-A(J)*GA(J-1))
      GA(J)=REC*C(J)
   10 OUT(J)=REC*(RHS(J)-A(J)*OUT(J-1))
      DO 20 JJ=2,N
      J=N-JJ+1
   20 OUT(J)=OUT(J)-GA(J)*OUT(J+1)
      RETURN
      END



      SUBROUTINE SOLV1
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),PI(31),AA(162),BB(162),CC(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELK,RA,DCN,DSN,RA4,EPSIL,CCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALOI
     5 ,SCALQO,N6,GAMMA,NOPT,CSTAR,FEM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,GPL,GPU
      COMPLEX FF,F1,GG
      DIMENSION CX(162),SX(162),FF(162),GG(162),F1(31)
      COMMON /SOL1/ Q(162,31)
      IF(NEW.NE.1) GO TO 30
      DO 1 I=1,M
      CX(I)=COS((I-1)*DT)
      SX(I)=SIN((I-1)*DT)
    1 CONTINUE
      NEW=0.
      MMP=MM+1
   30 CONTINUE
      MA=M/2
      MA1=MA+1
      DO 2 J=1,N,2
```

```
      CALL TWOFFT(M,Q(1,J),Q(1,J+1),FF,GG,CX,SX,1)
      DO 7 I=1,MA1
      IM=M-I+3
      Q(I,J)=REAL(FF(I))
      Q(I,J+1)=REAL(GG(I))
      Q(IM,J)=-AIMAG(FF(I))
      Q(IM,J+1)=-AIMAG(GG(I))
    7 CONTINUE
    2 CONTINUE
      HR=.5*DR
      DO 3 J=1,N
      D(J)=2.*RS(J)
      T=RA*RA*R(J)
      B(J)=T*(R(J)-HR)
    3 C(J)=T*(R(J)+HR)
      C(1)=D(1)
      DO 4 I=1,MA1
      IM=M-I+3
      DO 5 J=1,N
      A(J)=-D(J)-2.*(1.-CX(I))
    5 FF(J)=CMPLX(Q(I,J),Q(IM,J))
      CALL TRID1(B,A,C,FF,F1,N,162)
      DO 8 J=1,N
      Q(I,J)=REAL(F1(J))
      Q(IM,J)=AIMAG(F1(J))
    8 CONTINUE
    4 CONTINUE
      DO 9 J=1,N,2
      DO 10 I=1,MA1
      IM=M-I+3
      FF(I)=CMPLX(Q(I,J),-Q(IM,J))
      GG(I)=CMPLX(Q(I,J+1),-Q(IM,J+1))
      FF(IM)=CMPLX(Q(I,J),Q(IM,J))
   10 GG(IM)=CMPLX(Q(I,J+1),Q(IM,J+1))
      CALL TWOFFT(-M,Q(1,J),Q(1,J+1),FF,GG,CX,SX,1)
    9 CONTINUE
      DO 12 J=1,N
      Q(MM,J)=Q(1,J)
   12 Q(MMP,J)=Q(2,J)
      RETURN
      END


      SUBROUTINE SWEEP1
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,KN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,CCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSLP,SEPM,TTLE(4),M,N,MM,NN,NSP
```

```
    3 ,IK,JK,IZ,ITYP,MODE,I3,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
    4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
    5 ,SCALQO,NG,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
    6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      COMMON /SOL1/ Q(162,31)
      DATA Q/5022*0.0/
      YP=0.
      NSP=0
      DO 10 J=1,NN
      PHI(MM,J)=PHI(1,J)+DPHI
      PHI(MM+1,J)=PHI(2,J)+DPHI
   10 CONTINUE
      TE=-2
      DO 30 I=LL,MM
      CALL MURMAN1
      DO 100 J=1,N
      Q(I,J)=D(J)
  100 CONTINUE
   30 CONTINUE
      TE=2
      I=LL
   80 I=I-1
      CALL MURMAN1
      DO 60 J=1,N
      Q(I,J)=D(J)
   60 CONTINUE
      IF(I.GT.2) GO TO 60
      DO 61 J=1,N
   61 Q(1,J)=Q(MM,J)
  210 FORMAT(5(2I4,E16.8))
      CALL SOLV1
  200 FORMAT(5(I4,E16.8))
      DO 110 I=1,M
      DO 110 J=1,N
  110 PHI(I,J)=PHI(I,J)+Q(I,J)
      DO 111 J=1,N
  111 PHI(MM,J)=PHI(1,J)+DPHI
      IF(RCL.EQ.0.) GO TO 90
      YA=RCL*((PHI(M,1)-(PHI(2,1)+DPHI))*DELTH+S1(1))
      IF(MODE.EQ.1) GO TO 90
      IF (NDES.GE.0) GO TO 41
      ALP=ALP-.5*YA
      GO TO 42
   41 BB(1) = BB(1)-.5*YA
   42 CALL COSI
      GO TO 95
   90 YA=TP*YA/(1.+BET)
      DPHI=DPHI+YA
   95 DO 97 L=1,M
   97 PHI(L,NN)=DPHI*PHIR(L)
      IF(MODE.EQ.0) RETURN
      DO 120 J=1,N
      DO 120 L=1,M
  120 PHI(L,J)=PHI(L,J)+YA*PHIR(L)
```

```
      RETURN
      END



      SUBROUTINE MURMAN1
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XF,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DP,DELTH,DELR,RA,CCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NOPT,CSTAR,REM,DEP,QINF,ISTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      PHIO=PHI(I,2)-2.*DR*CO(I)
      PHIYP=PHI(I,2)-PHI(I,1)
      PHIYY=PHIYP+PHIO-PHI(I,1)
      PHIXX=PHI(I+1,1)+PHI(I-1,1)-PHI(I,1)-PHI(I,1)
      PHIXM=PHI(I+1,1)-PHI(I-1,1)
      PHIXP=PHI(I+1,2)-PHI(I-1,2)
      IF(I.NE.MM) GO TO 10
      D(1)=C1*(PHIXX+RS(1)*PHIYY+RA4*CO(1))
      D(1)=-D(1)/C1
      GO TO 40
   10 U=PHIXM*DELTH-SI(I)
      BQ=U/FP(I,1)
      QS=U*BQ
      J=1
      IF(QS.LE.QCRIT) GO TO 30
      D(1)=0.
      GO TO 40
   30 CONTINUE
      CS=C1-C2*QS
      BQ=BQ*CS*(FP(I-1,1)-FP(I+1,1))
      X=RA4*(CS+QS)*CO(I)
      CMQS=CS-QS
      D(1)=CS*RS(1)*PHIYY+RI(1)*BQ+X+CMQS*PHIXX
      D(1)=-D(1)/CS
   40 CONTINUE
      DO 60 J=2,N
      PHIXX=PHI(I+1,J)+PHI(I-1,J)-PHI(I,J)-PHI(I,J)
      DU=PHIXP
      PHIXP=PHI(I+1,J+1)-PHI(I-1,J+1)
      PHIXY=PHIXP-PHIXM
      PHIXM=DU
      DU=DU*DELTH
      PHIYM=PHIYP
      PHIYP=PHI(I,J+1)-PHI(I,J)
      PHIYY=PHIYP-PHIYM
```

```
      U=R(J)*DU-SI(I)
      DV=R(J)*(PHI(I,J+1)-PHI(I,J-1))*DELR
      V=DV*R(J)-CO(I)
      RAV=R(J)*RA*V
      BQ=1./FP(I,J)
      BQU=BQ*U
      US=BQU*U
      UV=(BQU+BQU)*V
      VS=BQ*V*V
      QS=US+VS
      IF(QS.LE.QCRIT) GO TO 50
      D(J)=0.
      GO TO 60
   50 CS=C1-C2*QS
      CMVS=CS-VS
      CMUS=CS-US
      UV1=.5*BQU*RAV
      C(J)=RS(J)*CMVS
      D(J)=RA4*((CMVS+US-VS)*DV-UV*DU)+RI(J)*QS*BQ*(U*(FP(I-1,J)-FP(I+1,
     1J))+RAV*(FP(I,J-1)-FP(I,J+1)))+CMUS*PHIXX-UV1*PHIXY+C(J)*PHIYY
      D(J)=-D(J)/CS
   60 CONTINUE
      RETURN
      END



      SUBROUTINE TWOFFT(NS,F,G,ALP,BET,CN,SN,IDIM)
C     ABS(NS) IS THE NUMBER OF POINTS IN EACH ARRAY
C     DO FFT FOR F AND G OR REVERSE TRANSFORM FOR ALP AND BET
C     IF NS<0 THE REVERSE TRANSFORM IS PREFORMED
C     FUNCTIONS F AND G ARE REPRESENTED BY ARRAYS OF THEIR VALUES
C     ALP AND BET ARE COMPLEX FOURIER COEFFICIENTS FOR F AND G
C     ALP(N) IS OF THE FORM A(N)-I*B(N)
C     CN AND SN ARE THE COSINE AND SINE ARRAYS
C     IDIM IS THE SKIP FACTOR BETWEEN POINTS IN F AND G
      COMPLEX ALP,BET,X
      DIMENSION F(IDIM,1),G(IDIM,1),ALP(1),BET(1),CN(1),SN(1)
      N = IABS(NS)
      L = N/2
C     SET UP AND DO COMPLEX TRANSFORM
      IF (NS.LT.0) GO TO 20
      DO 10 J = 1,N
   10 ALP(J) = CMPLX(F(1,J),G(1,J))
      GO TO 40
C     SET UP FOR REVERSE TRANSFORM
   20 J=N+1
      DO 30 K = 1,L
      X =-CMPLX(AIMAG(BET(K))-REAL(ALP(K)),AIMAG(ALP(K))+REAL(BET(K)))
      ALP(J)=X
      X = CMPLX(REAL(ALP(K))+AIMAG(BET(K)),AIMAG(ALP(K))-REAL(BET(K)))
      ALP(K)=X
   30 J = J-1
```

```
      K=L+1
      ALP(K) =1.*(CMPLX(REAL(ALP(K))-AIMAG(BET(K)),AIMAG(ALP(K))-REAL(BE
     1T(K))))
   40 CALL FFORM(N,ALP,BET,CN,SN)
C     NOW SEPARATE OUT THE REAL AND IMAGINARY PARTS
      J = N
      IF (NS.LT.0) GO TO 60
      ENI=.5
      DO 50 K = 1,L
      X = CONJG(ALP(J))-ALP(K+1)
      BET(K+1) =-ENI*CMPLX( AIMAG(X),REAL(X))
      ALP(K+1) = ENI*(CONJG(ALP(K+1))+ALP(J))
   50 J = J-1
      BET(1) = (ENI+ENI)*AIMAG(ALP(1))
      ALP(1) = (ENI+ENI)*REAL(ALP(1))
      RETURN
   60 DO 70 J = 1,N
      F(1,J) = REAL(ALP(J))/N
   70 G(1,J) = -AIMAG(ALP(J))/N
      RETURN
      END
```

```
      FUNCTION VLAYER(EM2,A,B)
      X=(EM2-A)/(B-A)
      VLAYER = 0.
      IF (X.LE.0.) RETURN
      IF (X.GE.1.) GO TO 10
      VLAYER = 3.*X*X-2.*X*X*X
      RETURN
   10 VLAYER = 1.
      RETURN
      END
```



```
      SUBROUTINE NASHLS(K2)
C     QUADRATIC L.S. FIT H(K2-1),H(K2),SEPR(K2-1),SEPR(K2) USING
C     PREVIOUS 5 VALUES
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),CI(31),AA(162),BB(162),CD(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TF,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCFIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPSI,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      DIMENSION S(1),H(1),SEPR(1)
      EQUIVALENCE (S(1),FP(1,16)),(H(1),FP(1,6)),(SEPR(1),FP(1,14))
```

```
      F(S) = A111*S*S + B111*S + C111
      NLS = 5
      XNLS = FLOAT(NLS)
      LLS = NLS + 1
      X1 = 0.
      X2 = 0.
      X3 = 0.
      X4 = 0.
      DO 10 L=2,LLS
      DUM = S(K2-FLOAT(L*INC))
      X1 = X1 + DUM
      X2 = X2 + DUM*DUM
      X3 = X3 + DUM*DUM*DUM
   10 X4 = X4 + DUM*DUM*DUM*DUM
      Z211 = X2-X1*X1/XNLS
      Z321 = X3-X2*X1/XNLS
      Z422 = X4-X2*X2/XNLS
      Y1 = 0.
      Y2 = 0.
      Y3 = 0.
      DO 20 L=2,LLS
      DUM = S(K2-FLOAT(L*INC))
      DUMM = H(K2-FLOAT(L*INC))
      Y1 = Y1 + DUMM
      Y2 = Y2 + DUMM*DUM
   20 Y3 = Y3 + DUMM*DUM*DUM
      Z2 = Y2-Y1*X1/XNLS
      Z3 = Y3-Y1*X2/XNLS
      A111 = (Z3*Z211-Z2*Z321)/(Z422*Z211-Z321*Z321)
      B111 = (Z2-A111*Z321)/Z211
      C111 = (Y1-A111*X2-B111*X1)/XNLS
      H(K2-INC) = F(S(K2-INC))
      H(K2) = F(S(K2))
      Y1 = 0.
      Y2 = 0.
      Y3 = 0.
      DO 30 L=2,LLS
      DUM = S(K2-FLOAT(L*INC))
      DUMM = SEPR(K2-FLOAT(L*INC))
      Y1 = Y1 + DUMM
      Y2 = Y2 + DUMM*DUM
   30 Y3 = Y3 + DUMM*DUM*DUM
      Z2 = Y2-Y1*X1/XNLS
      Z3 = Y3-Y1*X2/XNLS
      A111 = (Z3*Z211-Z2*Z321)/(Z422*Z211-Z321*Z321)
      B111 = (Z2-A111*Z321)/Z211
      C111 = (Y1-A111*X2-B111*X1)/XNLS
      SEPR(K2-INC) = F(S(K2-INC))
      SEPR(K2) = F(S(K2))
      RETURN
      END
```

```
      SUBROUTINE INTPLI(MX,XI,FI,N,X,F,FP,FPP,FPPP)
      DIMENSION X(1),F(1),FP(1),FPP(1),FPPP(1),XI(1),FI(1)
      REAL NEW,LEFT
      DATA TOL /1.E-9 /
C     XI(L) WILL SATISFY F(XI(L)) = FI(L)  FOR L = 1 TO ABS(MX)
C     F,FP,FPP,FPPP ARE THE FUNCTION AND DERIVATIVES AT THE X POINTS
      NX =IABS(MX)
      K = 2
      L1 = 1
      NEW = X(1)
      FN = F(1)
      FVAL = FI(1)
      IF (ABS(F(1)-FI(1)).GT.TOL) GO TO 5
      L1 = 2
      XI(1) = X(1)
      IF (NX.EQ.1) RETURN
    5 DO 100 L = L1,NX
      IF ((FVAL.NE.FI(L)).OR.(L.EQ.1)) GO TO 6
      NEW = X(K)
      FN = F(K)
      IF (FP(K)*FP(K-1).GT.0.) GO TO 6
      ROOT = SQRT(FPP(K-1)**2-2.*FP(K-1)*FPPP(K-1))
      DX = -2.*FP(K-1)/(FPP(K-1)+SIGN(ROOT,FPP(K-1)))
      NEW = X(K-1)+DX
      FN =F(K-1)+DX*(FP(K-1)+DX*(.5*FPP(K-1)+DX*FPPP(K-1)/6.))
    6 FVAL = FI(L)
      SGN = F(K-1)-FVAL
      IF (NEW.GT.X(K-1)) SGN = FN-FVAL
      DO 10 J = K,N
      IF(FP(J)*FP(J-1).LE.0.) GO TO 7
      IF (SGN*(F(J)-FVAL).LE.0.) GO TO 20
      GO TO 10
    7 ROOT = SQRT(FPP(J-1)**2-2.*FP(J-1)*FPPP(J-1))
      DX = -2.*FP(J-1)/(FPP(J-1)+SIGN(ROOT,FPP(J-1)))
      RIGHT = X(J-1)+DX
      LEFT = AMAX1(X(J-1),NEW+TOL)
      IF (LEFT.GT.RIGHT) GO TO 10
      F2 = .5*FPP(J-1)
      F3 = FPPP(J-1)/6.
      FN    = F(J-1)+DX*(FP(J-1)+DX*(F2+DX*F3))
      IF (SGN*(FN-FVAL).LE.0) GO TO 65
   10 CONTINUE
      IF (MX.GT.0) GO TO 11
      MX = L-1
      RETURN
   11 PRINT 499,L,FI(L)
  499 FORMAT ( * TROUBLE AT *,I5,3X,F16.6)
      J = K
      GO TO 100
   20 OLD = AMAX1(X(J-1),NEW+TOL)
      F2 = .5*FPP(J-1)
      F3 = FPPP(J-1)/6.
      START=OLD
      DO 40 K = 1,10
```

```
      DX = OLD-X(J-1)
      FPOLD = FP(J-1)+DX*(FPP(J-1)+.5*DX*FPPP(J-1))
      IF (ABS(FPOLD).LE.TOL) GO TO 60
      FN = F(J-1)+DX*(FP(J-1)+DX*(F2+DX*F3))
      NEW = OLD-(FN-FVAL)/FPOLD
      IF (NEW.LT.START) GO TO 60
      NEW = AMIN1(NEW,X(J))
      IF (ABS(NEW-OLD).LT.TOL) GO TO 90
   40 OLD = NEW
      CALL ABORT
   60 RIGHT = X(J)
      LEFT = OLD
      IF (SGN*(FN-FVAL).GT.0.) GO TO 65
      RIGHT = LEFT
      LEFT = XI(L-1)
      IF (L.EQ.1) LEFT = X(1)
   65 DO 70  K = 1,50
      IF ((RIGHT-LEFT).LE.TOL) GO TO 90
      NEW = .5*(LEFT+RIGHT)
      DX = NEW-X(J-1)
      FN    = F(J-1)+DX*(FP(J-1)+DX*(F2+DX*F3))
      IF ((FN-FVAL)*SGN.LE.0.) GO TO 80
      LEFT= NEW
      GO TO 70
   80 RIGHT = NEW
   70 CONTINUE
   90 XI(L) = NEW
  100 K = J
      MX = NX
      RETURN
      END


      SUBROUTINE READQS
C     SUBROUTINE TO READ IN THE INPUT PRESSURE DISTRIBUTION
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIP(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NI,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NOPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      DIMENSION QI(1),SF(1),QX(1),SX(1),ES(1),GP(1),GPP(1),GPPP(1)
     1 ,DQDS(1),PHT(1),DPHDS(1),QPP(1),QPPP(1),Q(1)
      EQUIVALENCE (QI(1),FP(1,1)),(SF(1),FP(1,3)),(QX(1),FP(1,5))
```

```
     1 ,(SX(1),FP(1,7)),(ES(1),FP(1,9)),(GP(1),FP(1,11))
     2 ,(GPP(1),FP(1,13)),(GPPP(1),FP(1,15)),(DQDS(1),FP(1,17))
     3 ,(PHT(1),FP(1,19)),(DPHDS(1),FP(1,21)),(QPP(1),FP(1,23))
     4 ,(QPPP(1),FP(1,25)),(Q(1),FP(1,27))
      XMACH(QS) = SQRT(QS/(.5*(GAMMA+1.)*CSTAR*CSTAR-.5*(GAMMA-1.)*QS))
      DATA NINMAX /300/
      MODE = 0
      REWIND N6
      READ(N6,500) XIN,CSTAR
  500 FORMAT (2F10.4)
      NIN = ABS(XIN)
      CALL GOPLCT(NRN)
      IF (NIN.GT.NINMAX) GO TO 90
C     READ IN Q(S)
      QMAX = 0.
      DO 20 J = 1,NIN
      READ(N6,510) SF(J),QI(J)
  510 FORMAT(2E20.10)
      QMAX = AMAX1(QMAX,ABS(QI(J)))
   20 CONTINUE
      XFAC = 2./(SF(NIN)-SF(1))
      CONST = -1.-XFAC*SF(1)
      FAC = 1.
      IF(QMAX.GT.1.6) FAC = 1.6/QMAX
      WRITE (N4,130)
      DO 30 J = 1,NIN
      SX(J) = XFAC*SF(J)+CONST
      QX(J) = FAC*QI(J)
      Q(J) =       XMACH(QX(J)*QX(J))
      WRITE (N4,140) J,SF(J),QI(J),SX(J),QX(J),Q(J)
      QI(J)=QX(J)
   30 CONTINUE
      WRITE (N4,600) CSTAR
  600 FORMAT (1X,/,5X,24HINPUT CRITICAL SPEED IS ,F10.4)
      CSTAR = FAC*CSTAR
      SIZE = .14
      SCX = 5.
      SCY = 2.5
      XOR = 6.0
      IF (XIN.LT.0.) XOR = 5.75
      CALL PLOT(XOR,5.5,-3)
      DO 60 L=1,NIN
   60 CALL SYMBOL(SCX*SX(L),SCY*QX(L),.5*SIZE,3,0.,-1)
      CALL PLOT(-5.0,-4.0,3)
      CALL PLOT(-5.0,4.0,2)
      CALL SYMBOL(-4.5,3.5,SIZE,1HQ,0.,1)
      CALL SYMBOL (-5.0,SCY*CSTAR,2.*SIZE,15,0.,-1)
      CALL SYMBOL (-5.0,-SCY*CSTAR,2.*SIZE,15,0.,-1)
      DO 70 L=1,9
      YH = FLOAT(L-1)-4.0
      S = .4*FLOAT(L-1)-1.6
      CALL SYMBOL(-5.0,YH,SIZE,15,0.,-1)
      ENCODE(10,100,A) S
   70 CALL SYMBOL(-5.7,YH,SIZE,A,0.,4)
```

```
      CALL PLOT(-5.0,0.,3)
      CALL PLOT(5.0,0.,2)
      CALL SYMBOL (4.5,-.5,SIZE,1HS,0.,1)
      DO 60 L=1,11
      XH = FLOAT(L-1)-5.0
      S = .2*FLOAT(L-1)-1.0
      CALL SYMBOL (XH,0.,SIZE,15,90.,-1)
      ENCODE (10,100,A) S
   60 CALL SYMBOL(XH-2.*SIZE,-.3,SIZE,A,0.,4)
  100 FORMAT(F4.1)
      CALL SYMBOL(-1.5,-4.5,SIZE,24HINPUT SPEED DISTRIBUTION,0.,24)
      SX(NIN) = 1.
      DS = (SX(NIN)-SX(1))/FLOAT(NQPT-1)
C     FIND Q(S) AT EVENLY SPACED POINTS
      ES(1) = SX(1)
      DO 40 J = 2,NQPT
   40 ES(J) = ES(J-1)+DS
      ES(NQPT) = SX(NIN)
      CALL SPLIF(NIN ,SX,QX,GP,GPP,GPPP,3,0.,3,0.)
      CALL INTPL(NQPT,ES,Q,SX,QX,GP,GPP,GPPP)
      CALL PLOT(SCX*ES(1),SCY*Q(1),3)
      DO 95 L=2,NQPT
   95 CALL PLOT(SCX*ES(L),SCY*Q(L),2)
      CALL PLOT(-XOR,-5.5,-3)
      CALL FRAME
      IF (CSTAR.LT.0.) GO TO 210
      CALL SPLIF(NQPT,ES,Q,GP,GPP,GPPP,3,0.,3,0.)
      CALL INTPLI(1,SO,0.,NQPT,ES,Q,GP,GPP,GPPP)
C     INTEGRATE Q(S) TO GET PHI(S)
      CALL SPLIF(NQPT,ES,Q,DQDS,QPP,PHT,-3,0.,3,0.)
      CALL INTPL(1,SO,PHMN,ES,PHT,Q,DQDS,QPP)
      GAM = PHT(NQPT)-PHT(1)
      SCALQI = PHT(NQPT)-PHMN
      DO 50 I = 1,NQPT
      QI(I) = PHT(I)-PHMN
      VAL = AMAX1(0.,QI(I))/SCALQI
   50 PHT(I) = SIGN(SQRT(VAL),ES(I)-SO)
      REWIND N6
      WRITE (N6) (PHT(J),ES(J),QI(J),Q(J),J=1,NQPT)
      CALL INCOMP
      RETURN
   90 WRITE (N4,110) NINMAX
  210   CALL PLOT(0.,0.,999)
      CALL EXIT
      RETURN
  110 FORMAT (15H0****MORE THAN ,I4,30H INPUT CARDS NOT PERMITTED**** /
     1 32HC***PROGRAM STOPPED IN READQS*** )
  130 FORMAT(1H0/11X,4HCARD,5X,7HS-INPUT,8X,7HQ-INPUT,10X,6HS-USED
     1 ,9X,6HQ-USED, 11X,6HM-USED /)
  140 FORMAT (3X,I9,2F15.6,3X,2F15.6,3X,F15.6)
      END
```

```
      SUBROUTINE INCOMP
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      QFAC = SCALQI-.5*GAM
      DO 10 I=1,10
      TAU = ASIN(-GAM/(PI*QFAC))
   10 QFAC = (SCALQI+GAM*TAU/PI-.5*GAM)/COS(TAU)
      TAU = ASIN(-GAM/(PI*QFAC))
      BB(1) = -TAU-ALP
      DPHI = 4.*GAM/QFAC
      CALL COSI
      ANG = 0.
      DO 20 I=1,M
      PHI(I,1) = (.5*QFAC-1.)*CO(I)+GAM*ANG/TP
      ANG = ANG+DT
      PHI(MM,1) = PHI(1,1)+GAM
      PHI(MM+1,1) = PHI(2,1)+GAM
      INC = 1
      CALL CYCLE
      RETURN
      END
```

```
      SUBROUTINE CYCLE
      COMMON/FL/FLUXT4,CC4,CDW,INDCD
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),RPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
      DIMENSION PHIS(1),CIRC(1),DPHDW(1),D2PHDW(1),PHIT(1),Q(1)
     1 ,FPP(1),FPPP(1),FPPPP(1),QX(1),PHIV(1),DS(1),CX(1),SX(1),SS(1)
     2 ,CCP(1),A1(1),A2(1),A3(1),A4(1),B1(1)
      EQUIVALENCE (  DS(1),FP(1,1)),(CIRC(1),FP(1,3)),(DPHDW(1),FP(1,5))
     1 ,(D2PHDW(1),FP(1,7)),(PHIT(1),FP(1,9)),(Q(1),FP(1,11))
```

```
     2 ,(FPP(1),FP(1,13)),(FPPP(1),FP(1,15)),(FPPPP(1),FP(1,17))
     3 ,(QX(1),FP(1,19)),(PHIV(1),FP(1,21)),(PHIS(1),FP(1,23))
     4 ,(CX(1),FP(1,25)),(SX(1),FP(1,27)),(SS(1),FP(1,29))
     5 ,(CCP(1),PHIR(1)),(A1(1),RP(1)),(A2(1),RP(7))
     6 ,(A3(1),RP(13)),(A4(1),RP(19)),(B1(1),RP(25))
      XMACH(QS) = SQRT(QS/(.5*(GAMMA+1.)*CSTAR*CSTAR-.5*(GAMMA-1.)*QS))
      CP(Q) = C5*((C4/(1.+C2*Q*Q))**C7-1.)
      DATA KD/0/
      LC = NFC
      NMP = 2*LC
      MC = NMP +1
      PILC = PI/FLOAT(LC)
      NDUM1 = NMP/M
      NDUM2 = NDUM1-1
      IF (INC.EQ.1) GO TO 6080
      INDCD = 1
      CALL GTURB(0.,0.,.4,CDW,0.,.125,.1)
      INDCD = 0
 6080 DO 10 I=1,MM
      CIRC(I) = FLOAT(I-1)*DT
   10 PHIS(I) = PHI(I,1)+CO(I)
      BB1 = BB(1)
      CALL SPLIF(MM,CIRC,PHIS,DPHDW,  FPPP, FPPPP,1,0.,1,0.)
      IF (MM.EQ.MC) GO TO 5
      DO 6 I=1,MM
      Q(I) = PHIS(I)
    6 SS(I) = CIRC(I)
      DO 7 I=1,MC
    7 CIRC(I) = FLOAT(I-1)*PILC
      CALL INTPL(MC,CIRC,PHIS,SS,Q,DPHDW,FPPP,FPPPP)
      CALL SPLIF(MC,CIRC,PHIS,DPHDW,  FPPP, FPPPP,1,0.,1,0.)
    5 DO 9 I=41,120
      SS(I-40) = CIRC(I)
    9 Q(I-40) = DPHDW(I)
      CALL SPLIF(80,SS,Q,FPP,FPPP,FPPPP,3,0.,3,0.)
      CALL INTPLI(1,WNP,0.,80,SS,Q,FPP,FPPP,FPPPP)
      CALL SPLIF(MC,CIRC,PHIS,DPHDW,D2PHDW,FPPP,1,0.,1,0.)
      CALL INTPL(1,WNP,PHMN,CIRC,PHIS,DPHDW,D2PHDW,FPPP)
      SCALQD = PHIS(MC)-PHMN
      REWIND N6
      READ (N6) (PHIT(I),SS(I),PHIV(I),Q(I),I=1,NQPT)
      CALL SPLIF(NQPT,PHIT,Q,FPP,FPPP,FPPPP,3,0.,3,0.)
      VAL = SQRT((PHIS(1)-PHMN)/SCALQD)
      DO 20 I=1,MC
      VAL = AMAX1(0.,PHIS(I)-PHMN)/SCALQD
      FAC = 1.
      IF (CIRC(I).LE.WNP) FAC = -1.
   20 PHIS(I) = FAC*SQRT(VAL)
      CALL INTPL(MC,PHIS,QX,PHIT,Q,FPP,FPPP,FPPPP)
      IF (INC.EQ.1) GO TO 8887
C     DETERMINE EM
      XNUM = 0.
      DEN = 0.
      DO 4444 J = 2,M
```

```
      K = J
      IF (MC.NE.MM) K=NDUM1*J-NDUM2
      VAL = (PHI(J+1,1)-PHI(J-1,1))*DELTH-SI(J)
      VAL = (VAL*VAL)/FP(J,1)
      FPPP(J) = VAL
      IF ((QX(K)*QX(K)).GT.(CSTAR*CSTAR)) GO TO 4444
      XNUM = XNUM + QX(K)*QX(K)*VAL
      DEN = DEN + VAL*VAL
 4444 CONTINUE
      QINF = SQRT(XNUM/DEN)
      DQMAX = 0.
      DQAVE = 0.
      DO 4445 J = 2,M
      K = J
      IF (MC.NE.MM) K=NDUM1*J-NDUM2
      DQAVE  = DQAVE  + ((QX(K)*QX(K))-QINF*QINF*FPPP(J))*
     1 ((QX(K)*QX(K))-QINF*QINF*FPPP(J))
      DQMAX = AMAX1(DQMAX,ABS(SQRT(FPPP(J))-(ABS(QX(K))/QINF)))
 4445 CONTINUE
      DQAVE = SQRT(DQAVE/FLOAT(M-1))
      C2 = .5*(GAMMA-1.)
      C7 = GAMMA/(GAMMA-1.)
      EMX = EM
      EM = (GAMMA+1.)*CSTAR*CSTAR/(QINF*QINF)-GAMMA+1.
      EM = 2./EM
      EM = SQRT(EM)
      EM = (1.-REM)*EM+REM*EMX
      C1 = C2+1./(EM*EM)
      C6 = C2*EM*EM
      C4 = 1.+C6
      C5 = 1./(C6*C7)
      QCRIT = (C1+C1)/(GAMMA+1.)
C     DETERMINE SCALING FOR PHI BY LEAST SQUARES FIT
      CALL SPLIF(NQPT,PHIT,PHIV,FPP,FPPP,FPPPP,3,0.,3,0.)
      CALL INTPL(MC,PHIS,Q ,PHIT,PHIV,FPP,FPPP,FPPPP)
      Q(MC) = PHIV(NQPT)
      XNUM = 0.
      DEN = 0.
      DO 47 J=1,MM
      K = J
      IF (MC.NE.MM) K=NDUM1*J-NDUM2
      XNUM = XNUM+PHI(J,1)+CO(J)-PHMN
   47 DEN = DEN+ Q(K)
      FAC = XNUM/DEN
      DPHI = FAC*GAM
 8887 DO 8886 J=1,MM
      K = J
      IF (MC.NE.MM) K=NDUM1*J-NDUM2
 8886 CCP(J) = QX(K)
      CALL SPLIF(NQPT,PHIT,SS,FPP,FPPP,FPPPP,3.,0.,3.,0.)
      CALL INTPL(MC,PHIS,SX,PHIT,SS,FPP,FPPP,FPPPP)
C***************************
      CALL SPLIF(MC,CIRC,SX,DS,FPPP,FPPPP,1,C.,1,0.)
      DO 40 I=2,NMP
```

```
      VAL = ABS(DS(I)/(2.*SIN(CIRC(I)*.5)))
      DS(I) = ALOG(VAL)
   40 CONTINUE
C***************************
      VAL = ABS(FPPP(1))
      DS(1) = ALOG(VAL)
      DS(MC) = DS(1)
   43 CONTINUE
      DO 240 I=1,MC
      ANGL = FLOAT(I-1)*PILC
      CX(I) = COS(ANGL)
  240 SX(I) = SIN(ANGL)
      DO 1040 I=1,LC
      FPPP(I) = AA(I)
      FPPPP(I) = BB(I)
      AA(I) = CX(2*I-1)
 1040 BB(I) = -SX(2*I-1)
      CALL FOUCF(NMP,DS,FPP,AA,BB)
      DFAVE = 0.
      DO 1050 I=1,LC
      AA(I) = -AA(I)
      IF (FPPP(I).EQ.99999.) GO TO 1050
      AA(I) = TSTEP*AA(I) + (1.-TSTEP)*FPPP(I)
      BB(I) = TSTEP*BB(I) + (1.-TSTEP)*FPPPP(I)
 1050 DFAVE  = DFAVE +(AA(I)-FPPP(I))*(AA(I)-FPPP(I))+(BB(I)-FPPPP(I))*
     1 (BB(I)-FPPPP(I))
      DFAVE = SQRT(DFAVE/160.)
      BB(1) = BB1
      IF (INC.EQ.0) GO TO 45
      QINF = QFAC/(4.*EXP(AA(1)))
      EM = (GAMMA+1.)*CSTAR*CSTAR/(QINF*QINF)-GAMMA+1.
      EM = SQRT(2./EM)
      C2 = .5*(GAMMA-1.)
      C7 = GAMMA/(GAMMA-1.)
      C1 = C2+1./(EM*EM)
      C6 = C2*EM*EM
      C4 = 1.+C6
      C5 = 1./(C6*C7)
      QCRIT = (C1+C1)/(GAMMA+1.)
   45 CONTINUE
      IF (INC.EQ.1) GO TO 6030
      IF (KDES.EQ.1) GO TO 6040
      IF ((MOD( KD,KDES ).NE.1).AND.(KD.GE. 2)) GO TO 6000
 6040 WRITE (N2,6010)
 6010 FORMAT (1H1,/,8X,5HDQAVE,6X,5HDQMAX,6X,5HDFAVE,6X,4HDPHI,7X
     1 ,4HDBB1,5X,3HNSP,4X,2HEM,7X,2HCL,7X,3HALP,6X,4HANGO
     2 ,5X,3HCDW,6X,2HTC,/)
 6000 ANGO = -RAD*BB(1)
      ALP1 = ALP*RAD
      WRITE(N2,6020) KD,DQAVE,DQMAX,DFAVE,YR,YA,NSP,EM,CL,ALP1,ANGO
     1 ,CDW,TC
 6020 FORMAT (1X,I3,5(2X,F9.3),2X,I3,2X,F5.4,3(2X,F7.4)
     1 ,2X,F8.5,2X,F5.3)
 6030 INC = 0
```

```
      KD = KD+1
      IF (KD.GT.NDES) KD=1
      CALL MAP
      CL = 2.*DPHI*CHD
      REWIND M4
      WRITE(M4,800) MM
  800 FORMAT(10X,I3)
      SNX = 1.
      WRITE (M4,805)        TC,DQAVE ,YR,SNX
  805 FORMAT  (F7.3,2E10.2,F4.1)
      DO 5555 I = 1,MM
      VAL = XMACH(CCP(I)*CCP(I))
      CCP(I) = CP(VAL)
      WRITE(M4,810) XC(I),CCP(I)
 5555 CONTINUE
  810 FORMAT(2F10.4)
      CALL COSI
      EPS1 = EPS1-DEP
      RETURN
      END
```

```
      SUBROUTINE OUTPT
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),FPP(31),R(31),RS(31),FI(31),AA(162),BB(162),CO(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,FN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,RA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,PIT,BETA,FSYM,XSEP,SEPM,ITLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALDI
     5 ,SCALDO,NS,GAMMA,NOPT,CSTAR,FLM,DLP,QINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,CPL,CPU
      DIMENSION Q(1),CIRC(1),FPP(1),FPPP(1),FPPPP(1),FPPPPP(1),PHIS(1)
      EQUIVALENCE  (Q(1),FP(1,13)),(CIRC(1),FP(1,3)),(FPP(1),FP(1,5))
     1 ,(FPPP(1),FP(1,7)),(FPPPP(1),FP(1,9)),(FPPPPP(1),FP(1,11))
     2 ,(PHIS(1),FP(1,15))
      REWIND N3
      XM = MM
      IF (XOUT.GT.0.) GO TO 100
  100 QQ= SQRT(QCRIT)
      WRITE(N3,120) XM,QQ
  120 FORMAT(2F10.4)
      DO 140 L=2,M
      U = (PHI(L+1,1)-PHI(L-1,1))*DELTH-SI(L)
      QS = (U*U)/FP(L,1)
      Q(L) = SQRT(QS)
      PHIS(L-1) = PHI(L,1)+CO(L)
      CIRC(L-1) = FLOAT(L-1)*DT
  140 CONTINUE
```

```
      Q(1) = .5*(Q(2)+Q(M))
      Q(MM) = Q(1)
      MZ = M-1
      CALL SPLIF(MZ,CIRC,PHIS,FPP,FPPP,FPPPP,3,0.,3,0.)
      CALL SPLIF(MZ,CIRC,FPP,FPPP,FPPPP,FPPPPP,3,0.,3,0.)
      CALL INTPLI(1,WNP,0.,MZ,CIRC,FPP,FPPP,FPPPP,FPPPPP)
      Q(1) = -Q(1)
      DO 200 I=2,M
      Q(I) = SIGN(Q(I),CIRC(I-1)-WNP)
  200 CONTINUE
C     Q(MM) = Q(MM)
      ARC1 = 0.
      WRITE(N3,160) ARC1,Q(1)
      PRINT 160, ARC1,Q(1)
      DO 150 L=2,MM
      DX = XC(L)-XC(L-1)
      DY = YC(L)-YC(L-1)
      ARC1 = ARC1 + SQRT(DX*DX+DY*DY)
      WRITE(N3,160) ARC1,Q(L)
      PRINT 160, ARC1,Q(L)
  160 FORMAT(2E20.10)
  150 CONTINUE
      XOUT = 0.
      RETURN
      END


      BLOCK DATA
      COMMON PHI(162,31),FP(162,31),A(31),B(31),C(31),D(31),E(31)
     1 ,RP(31),FPP(31),R(31),RS(31),RI(31),AA(162),BB(162),CC(162)
     2 ,SI(162),PHIR(162),XC(162),YC(162),FM(162),ARCL(162),DSUM(162)
     3 ,ANGOLD(162),XOLD(162),YOLD(162),ARCOLD(162),DELOLD(162)
     4 ,RP4(31),RP5(31)
      COMMON /A/ PI,TP,RAD,EM,ALP,RN,PCH,XP,TC,CHD,DPHI,CL,RCL,YR
     1 ,XA,YA,TE,DT,DR,DELTH,DELR,FA,DCN,DSN,RA4,EPSIL,QCRIT,C1,C2
     2 ,C4,C5,C6,C7,BET,BETA,FSYM,XSEP,SEPM,TTLE(4),M,N,MM,NN,NSP
     3 ,IK,JK,IZ,ITYP,MODE,IS,NFC,NCY,NRN,NG,IDIM,N2,N3,N4,NT,IXX
     4 ,NPTS,LL,I,LSEP,M4,NEW,EPS1,NDES,XLEN,SCALQI
     5 ,SCALQO,N6,GAMMA,NQPT,CSTAR,REM,DEP,WINF,TSTEP,XOUT
     6 ,INC,QFAC,GAM,KDES,PLTSZ,QPL,QPU
C     ****IDIM MUST BE SET TO THE FIRST DIMENSION OF PHI****
      DATA PI/3.14159265358979/ , EM/.75/ ,  ALP/0./ ,  CL/100./ ,
     1      PCH/.07/ , FSYM/1.0/ , RCL/1.0/ , BETA/0.0/, RN/20.00/ ,
     2 SEPM/.004/ , XSEP/.93/ , XP/0.0/ , M/160/ , N/30/ , NRN/1/ ,
     3 NFC/80/ , NPTS/81/ , LL/0/ , NG/1/ , IS/2/ , IDIM/162/ , MODE /1/
     4  , JK/0/ , N2/2/ , N3/3/ , N4/4/ , LSEP/161/ , IZ/125/ , ITYP/1/
     5 ,NQPT/321/,REM/0./,EPS1/0./,CSTAR/100./
     6 ,DEP/0./,NDES/-1/,TSTEP/.2/,KDES/10/,PLTSZ/50./,QPL/.65/,
     7 QPU/.95/
      END
```

END

DATE

FILMED

SEP 27 1979

End of Document